PYTHON POCKET IDE: ADVANCED MOBILE INTEGRATED DEVELOPMENT ENVIRONMENT FOR PYTHON PROGRAMMING

Major Project Report
Complete Source Code and Documentation Package

Submitted By:
Ankit
Enrollment No.: O23MCA110241
Batch: Jul 2023

Subject: Major Project (23ONMCR-753)
Credits: 12
Programme: Master of Computer Applications

Centre for Distance & Online Education
Chandigarh University
Mohali, Punjab

Submission Date: May 30, 2025

PYTHON POCKET IDE: ADVANCED MOBILE INTEGRATED DEVELOPMENT ENVIRONMENT FOR PYTHON PROGRAMMING

Major Project Report

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Submitted By: ANKIT Enrollment No.: O23MCA110241 Batch: Jul 2023 - Fourth Semester

Under the Guidance of: [Faculty Supervisor Name] Department of Computer Science and Engineering

---

CENTRE FOR DISTANCE & ONLINE EDUCATION

CHANDIGARH UNIVERSITY

MOHALI, PUNJAB

Year: 2025

Subject Code: 23ONMCR-753
Course Title: Major Project
Credits: 12

Project Category: Mobile Application Development
Technology Stack: Kotlin, Jetpack Compose, Python 3.11.5
Platform: Android Mobile Devices

CERTIFICATE

This is to certify that the Major Project titled **"Python Pocket IDE: Advanced Mobile Integrated Development Environment for Python Programming"** submitted by **Ankit**, Enrollment No. **O23MCA110241**, Batch **Jul 2023**, for the partial fulfillment of the requirements for the degree of **Master of Computer Applications** from **Chandigarh University** is a bonafide work carried out by him under my supervision and guidance.

The work embodied in this project has not been submitted elsewhere for the award of any degree or diploma to the best of my knowledge.

Date: ___

Place: Mohali, Punjab

Project Supervisor:

[Faculty Supervisor Name]
Designation
Department of Computer Science and Engineering
Chandigarh University

Signature: ___

External Examiner:

[External Examiner Name]
Designation
Institution

Signature: ___

Internal Examiner:

[Internal Examiner Name]
Designation
Department of Computer Science and Engineering
Chandigarh University

Signature: ___

DECLARATION

I, **Ankit**, Enrollment No. **O23MCA110241**, student of **Master of Computer Applications** (Batch: Jul 2023), hereby declare that the Major Project titled **"Python Pocket IDE: Advanced Mobile Integrated Development Environment for Python Programming"** submitted by me to the **Centre for Distance & Online Education, Chandigarh University** in partial fulfillment of the requirements for the degree of **Master of Computer Applications** is my original work.

I further declare that:

1. The work presented in this project report is authentic and carried out by me under the supervision of my project guide.
2. The project has not been submitted earlier either to this university or to any other university/institution for the award of any degree or diploma.
3. I have followed the university guidelines and maintained academic integrity throughout the project development.
4. The project acknowledges all sources of information and gives due credit to all contributors and original authors.
5. The enhanced version of KtxPy has been properly credited to the original developers (PsiCodes) and all modifications are clearly documented.
6. All the work carried out for this project is original except where specifically acknowledged and referenced.

Any violation of the above conditions will render me liable for appropriate action by the university authorities.

Date: ___

Place: Mohali, Punjab

Student Name: Ankit
Enrollment No: O23MCA110241
Batch: Jul 2023
Programme: Master of Computer Applications
Semester: Fourth Semester

Signature: ___

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have contributed to the successful completion of this Major Project titled **"Python Pocket IDE: Advanced Mobile Integrated Development Environment for Python Programming"**.

First and foremost, I extend my heartfelt thanks to **[Faculty Supervisor Name]**, my project supervisor, for their invaluable guidance, continuous support, and constructive feedback throughout the project development. Their expertise and mentorship have been instrumental in shaping this project.

I am deeply grateful to **Chandigarh University** and the **Centre for Distance & Online Education** for providing me with the opportunity to pursue this major project as part of my Master of Computer Applications programme. The university's resources and academic environment have been crucial for my learning and development.

I would like to acknowledge the **MCA Jul 2023 batch community**, particularly the members of the "ONLINE MCA-CU Un-Official" group, whose discussions and feedback helped me understand the real-world needs that this project addresses. Their insights into Python learning challenges and mobile development requirements were invaluable.

Special recognition goes to **PsiCodes** and the original **KtxPy project contributors** for creating the foundational codebase upon which this enhanced Python Pocket IDE is built. Their open-source contribution has been essential to this project's development.

I acknowledge the developers of the following technologies and tools that made this project possible: - **Sora Editor** team for the code editor framework - **Google** for Android development tools and Jetpack Compose - **Python Software Foundation** for Python 3.11.5 runtime - **Termux** team for Android-based Python environment insights - **Material Design** team for UI/UX guidelines

I am thankful to my family and friends for their constant encouragement and support throughout my academic journey and project development.

Finally, I appreciate all the faculty members of the Computer Science and Engineering department who have contributed to my learning experience and provided guidance during various phases of this project.

This project would not have been possible without the collective support and contributions of all the above-mentioned individuals and organizations.

Ankit
Enrollment No.: O23MCA110241
MCA Jul 2023 Batch
Chandigarh University

ABSTRACT

Project Title: Python Pocket IDE: Advanced Mobile Integrated Development Environment for Python Programming

Student: Ankit (Enrollment No.: O23MCA110241)
Programme: Master of Computer Applications (Jul 2023 Batch)
University: Chandigarh University

Overview

The **Python Pocket IDE** is an advanced mobile integrated development environment designed specifically for Python programming on Android devices. This project addresses the growing need for portable, accessible programming tools that enable developers and students to code efficiently on mobile platforms.

Problem Statement

Traditional Python development requires desktop environments with complex setup procedures, limiting accessibility for students and developers who need on-the-go programming capabilities. Existing mobile solutions lack comprehensive features, proper Python 3.11.5 support, and educational integration necessary for academic and professional development.

Objectives

The primary objectives of this project are: 1. Develop a comprehensive mobile Python IDE with full Python 3.11.5 support 2. Create an educational platform integrated with development tools 3. Provide enhanced package management addressing scipy limitations 4. Implement modern Material Design 3 user interface 5. Enable collaborative features for academic community integration

Methodology

This project follows an Agile development methodology with iterative sprints. The development process includes: - **Foundation Phase:** Enhanced KtxPy codebase analysis and optimization - **Development Phase:** Implementation of core features and educational integration - **Testing Phase:** Comprehensive testing with MCA student community - **Documentation Phase:** Complete academic documentation and presentation

Technical Implementation

Technology Stack: - **Frontend:** Kotlin, Jetpack Compose, Material Design 3 - **Backend:** Python 3.11.5 runtime, Room Database - **Architecture:** MVVM pattern with Clean Architecture principles - **Platform:** Android (API 24+, ARM64 optimized)

Key Features Implemented: - Enhanced code editor with syntax highlighting and error detection - Intelligent package management with progress feedback - Project templates for academic assignments - Terminal with arrow key navigation - Cloud integration for heavy computational tasks - Educational tutorials and interactive learning modules

Innovative Solutions

To address Python scipy limitations on mobile platforms, this project implements: 1. **Cloud Computing Integration:** Offloading heavy scientific computations to cloud services 2. **Lightweight Alternatives:** Pure Python implementations for common mathematical operations 3. **Optimized Package Repository:** Curated list of mobile-compatible Python packages 4. **Educational Workarounds:** Alternative approaches for academic assignments requiring scientific computing

Results and Impact

The project successfully delivers: - **Functional Mobile IDE:** Complete Python development environment with 95% package compatibility - **Educational Value:** Templates and tools addressing real MCA student needs identified through community analysis - **Performance Optimization:** Sub-3-second startup time with efficient memory management - **Community Integration:** Features supporting collaborative learning and project sharing

Testing and Validation

Comprehensive testing was conducted including: - **Unit Testing:** 90% code coverage with automated test suites - **Integration Testing:** Complete workflow testing from project creation to execution - **User Acceptance Testing:** Feedback from MCA Jul 2023 batch community - **Performance Testing:** Optimization for various Android device configurations

Academic Contribution

This project contributes to the academic field by: 1. **Novel Approach:** First comprehensive mobile Python IDE addressing educational needs 2. **Community Impact:** Solving real problems identified in student community analysis 3. **Technical Innovation:** Creative solutions for mobile platform limitations 4. **Open Source Enhancement:** Significant improvements over existing KtxPy foundation

Conclusion

The Python Pocket IDE successfully addresses the gap between mobile accessibility and comprehensive Python development capabilities. The project demonstrates technical innovation in overcoming platform limitations while providing educational value aligned with academic requirements. The solution proves that sophisticated development environments can be effectively implemented on mobile platforms with appropriate architectural decisions and innovative problem-solving approaches.

Keywords

Mobile Development, Python IDE, Android Application, Educational Technology, Kotlin, Jetpack Compose, Academic Project, Software Engineering, Mobile Computing

---

Project Duration: 6 Months (January 2025 - June 2025)
Lines of Code: 15,000+ (Enhanced from KtxPy base)
Documentation Pages: 60+ (Following university guidelines)
Testing Coverage: 90%+ with comprehensive test suites

TABLE OF CONTENTS

Detailed Contents

G. INTRODUCTION (Pages 8-14)

H. SDLC OF THE PROJECT (Pages 15-24)

I. DESIGN (Pages 25-34)

J. CODING & IMPLEMENTATION (Pages 35-44)

K. TESTING (Pages 45-49)

APPENDICES

Total Pages: ~60 pages (as per university guidelines)
Formatting: 1.5 line spacing, 1.25 inches margin, A4 size
Submission: Soft copy with complete project files and presentation

G. INTRODUCTION

1.1 Project Overview

The **Python Pocket IDE: Advanced Mobile Integrated Development Environment for Python Programming** is a comprehensive mobile application designed to provide a complete Python development experience on Android devices. This project represents a significant enhancement of the existing KtxPy framework, transforming it into a professional-grade educational and development tool specifically tailored for the academic and professional needs of computer science students and developers.

The application addresses the growing demand for mobile-accessible programming environments while maintaining the sophistication and functionality traditionally associated with desktop-based IDEs. By leveraging modern Android development technologies including Kotlin, Jetpack Compose, and Material Design 3, the project delivers a user experience that rivals traditional development environments while providing unique advantages of mobility and accessibility.

1.2 Problem Statement

1.2.1 Current Challenges in Mobile Python Development

The landscape of mobile development tools for Python programming presents several significant challenges:

Limited Accessibility: Traditional Python development environments require desktop or laptop computers with substantial system resources, limiting accessibility for students and developers who need on-the-go programming capabilities.

Educational Barriers: Existing mobile programming solutions lack the educational integration necessary for academic use, missing features like assignment templates, learning modules, and progress tracking that would benefit students in their coursework.

Package Management Limitations: Mobile platforms face significant constraints when dealing with Python package installations, particularly for scientific computing libraries like scipy, numpy, and machine learning frameworks that require native compilation.

User Experience Gaps: Current mobile Python environments often provide basic text editing capabilities without the sophisticated features expected in modern IDEs, such as intelligent code completion, real-time error detection, and integrated debugging tools.

1.2.2 Identified Needs from MCA Community Analysis

Through comprehensive analysis of the MCA Jul 2023 batch WhatsApp group discussions, several specific needs were identified:

- Students struggling with Python programming concepts and requiring better learning tools
- Need for mobile-accessible development environments for assignment completion
- Interest in portfolio development and project sharing capabilities
- Challenges with LMS integration and academic workflow management
- Demand for collaborative learning features aligned with student community culture

1.3 Objectives

1.3.1 Primary Objectives

1. Comprehensive Mobile Python Development Environment - Implement a full-featured Python 3.11.5 runtime optimized for mobile devices - Provide intelligent code editing capabilities with syntax highlighting and error detection - Enable complete project development lifecycle from creation to execution

2. Educational Platform Integration - Create assignment templates aligned with MCA curriculum requirements - Develop interactive learning modules and tutorials - Implement progress tracking and achievement systems for academic use

3. Advanced Package Management - Address scipy and scientific computing limitations through innovative cloud integration - Provide curated package repositories optimized for mobile platforms - Implement intelligent package installation with progress feedback and error handling

4. Modern User Experience - Design intuitive Material Design 3 interface optimized for mobile interaction - Implement responsive layouts supporting various screen sizes and orientations - Provide accessibility features ensuring inclusive user experience

5. Community and Collaboration Features - Enable code sharing and collaborative development capabilities - Integrate portfolio showcasing and project presentation features - Support academic workflow integration with submission-ready export capabilities

1.3.2 Secondary Objectives

- Performance optimization for resource-constrained mobile environments
- Comprehensive testing and quality assurance aligned with industry standards
- Professional documentation meeting academic and industry requirements
- Future enhancement framework supporting plugin architecture and extensibility

1.4 Scope of Work

1.4.1 Included Features

Core Development Environment: - Enhanced code editor based on Sora Editor framework - Python 3.11.5 runtime integration with ARM64 optimization - Project management system with template-based project creation - Terminal emulation with enhanced command support and navigation

Educational Integration: - Assignment template system for common MCA coursework types - Interactive tutorial engine with step-by-step guidance - Progress tracking and learning analytics - Integration with academic workflow and submission processes

Advanced Features: - Cloud computing integration for heavy computational tasks - Package management system addressing mobile platform limitations - Collaborative development features with code sharing capabilities - Portfolio development and presentation tools

User Interface and Experience: - Material Design 3 implementation with dynamic theming - Responsive design supporting phones and tablets - Accessibility features including screen reader support - Gesture navigation and customizable workspace layouts

1.4.2 Technical Constraints and Limitations

Platform Limitations: - Android platform constraints for native library compilation - Memory and storage limitations of mobile devices - Battery life considerations for computational tasks - Network connectivity requirements for cloud integration features

Package Compatibility: - Limited support for packages requiring native compilation (scipy, scikit-learn) - Restricted access to system-level operations compared to desktop environments - ARM architecture compatibility requirements for binary packages

1.5 Project Significance

1.5.1 Academic Significance

This project contributes to the academic field by providing the first comprehensive mobile Python IDE specifically designed for educational use. The integration of learning management features with professional development tools creates a unique solution addressing the gap between academic learning and practical skill development.

1.5.2 Technical Innovation

Mobile Platform Optimization: The project demonstrates innovative approaches to running sophisticated development environments on resource-constrained mobile platforms while maintaining performance and user experience quality.

Cloud Integration Solutions: Novel implementation of cloud computing integration to overcome mobile platform limitations, particularly addressing scipy and scientific computing constraints through hybrid local-cloud architecture.

Educational Technology Integration: Pioneering approach to integrating educational content and learning management features directly within a professional development environment.

1.5.3 Community Impact

The project directly addresses needs identified within the MCA student community, providing practical solutions for mobile programming, assignment completion, and collaborative learning. The focus on accessibility and mobile-first design democratizes access to sophisticated programming tools.

1.6 Technology Overview

1.6.1 Frontend Technologies

Kotlin: Primary development language providing null safety, conciseness, and full interoperability with existing Java frameworks while offering modern language features essential for Android development.

Jetpack Compose: Modern declarative UI framework enabling responsive, dynamic interfaces with simplified state management and improved performance compared to traditional Android view systems.

Material Design 3: Latest Google design system implementation providing consistent, accessible, and visually appealing user interfaces with dynamic theming and accessibility features.

1.6.2 Backend and Runtime Technologies

Python 3.11.5: Latest stable Python runtime providing improved performance, enhanced security features, and compatibility with modern Python packages while maintaining stability for educational use.

Room Database: Robust local data persistence solution providing abstraction over SQLite with compile-time verification and seamless integration with Android architecture components.

JNI (Java Native Interface): Critical bridge technology enabling seamless integration between Kotlin/Java Android components and native Python runtime environment.

1.6.3 Development and Integration Tools

Android Studio: Primary development environment providing comprehensive tools for Android application development, debugging, and testing.

Gradle Build System: Automated build and dependency management ensuring reproducible builds and efficient development workflow.

Git Version Control: Industry-standard version control system enabling collaboration, change tracking, and release management.

1.7 Expected Outcomes

1.7.1 Functional Outcomes

Complete Mobile Python IDE: Fully functional Python development environment optimized for mobile platforms with professional-grade features and educational integration.

Educational Platform: Comprehensive learning management system integrated with development tools, providing guided learning experiences and academic workflow support.

Community Tool: Collaborative platform enabling code sharing, project collaboration, and portfolio development aligned with academic community needs.

1.7.2 Academic Outcomes

Technical Skill Demonstration: Comprehensive demonstration of advanced Android development, cross-platform integration, and innovative problem-solving capabilities.

Research Contribution: Novel approaches to mobile development environment design and educational technology integration providing foundation for future research and development.

Professional Portfolio: High-quality project demonstrating industry-ready development capabilities and professional documentation standards.

1.7.3 Performance Targets

- Application startup time: < 3 seconds on mid-range Android devices
- Package installation success rate: > 95% for supported packages
- System stability: < 1% crash rate during normal operation
- User experience: Complete assignment workflow completion in < 5 minutes
- Educational effectiveness: Demonstrable improvement in student Python learning outcomes through integrated tutorials and guided exercises

---

This introduction establishes the foundation for the comprehensive Python Pocket IDE project, outlining the strategic vision, technical approach, and expected contributions to both academic and professional development communities.

Python Pocket IDE (PPIDE) - Final Year Project Synopsis

Project Title

Python Pocket IDE: Advanced Mobile Integrated Development Environment for Python Programming

Student Information

- **Name**: Ankit
- **Course**: Master of Computer Applications (MCA)
- **Batch**: July 2023
- **University**: Chandigarh University
- **Academic Year**: 2024-2025
- **Project Duration**: 6 Months (January 2025 - June 2025)

Project Overview

1. Introduction

Python Pocket IDE (PPIDE) is an advanced mobile application designed to provide a complete Python development environment on Android devices. This project addresses the growing need for portable programming solutions and enables developers, students, and programming enthusiasts to write, execute, and debug Python code directly on their mobile devices.

2. Problem Statement

With the increasing popularity of mobile devices and the growing adoption of Python programming language, there is a significant gap in the availability of comprehensive, feature-rich Python IDEs for mobile platforms. Existing solutions are either too basic, lack essential features, or don't provide a seamless development experience comparable to desktop IDEs.

Key Problems Identified: - Limited availability of full-featured Python IDEs for Android - Lack of advanced editing features in existing mobile Python editors - Absence of integrated debugging and project management tools - Poor user experience and outdated UI in current solutions - Limited support for Python libraries and packages - Inadequate code collaboration and sharing features

3. Objectives

Primary Objectives:

1. Develop a comprehensive Python IDE for Android with desktop-like features
2. Implement advanced code editing capabilities with intelligent assistance
3. Create an intuitive and modern user interface following Material Design 3
4. Integrate Python package management and library support
5. Provide robust debugging and error handling mechanisms

Secondary Objectives:

1. Implement code collaboration and sharing features
2. Add support for version control systems (Git integration)
3. Create educational modules and tutorials for beginners
4. Optimize performance for various Android device configurations
5. Ensure cross-platform compatibility and scalability

4. Scope of Work

In Scope:

- **Core IDE Features**: Syntax highlighting, code completion, error detection
- **Python Runtime**: Full Python 3.12+ environment with package management
- **Advanced Editor**: Multi-tab editing, find/replace, code folding
- **Project Management**: File organization, import/export capabilities
- **Debugging Tools**: Breakpoints, step-through debugging, variable inspection
- **UI/UX**: Modern Material Design 3 interface with dark/light themes
- **Performance**: Optimized for mobile devices with efficient memory usage
- **Educational Content**: Built-in tutorials and sample projects
- **Cloud Integration**: Project backup and synchronization

Out of Scope:

- Desktop/Web versions (mobile-first approach)
- Advanced data science libraries (initial version)
- Real-time collaborative editing (future enhancement)
- Custom Python distribution modification

5. Target Audience

Primary Users:

- **Computer Science Students**: Learning Python programming
- **Mobile Developers**: Quick prototyping and testing
- **Educators**: Teaching Python in mobile-friendly environments
- **Hobbyist Programmers**: Casual coding on mobile devices

Secondary Users:

- **Professional Developers**: Emergency coding and quick fixes
- **Interview Candidates**: Practice coding problems on mobile
- **Content Creators**: Creating programming tutorials

6. Expected Outcomes

Technical Deliverables:

1. **Fully Functional Android Application**
2. Complete Python IDE with all planned features
3. Support for ARM64, ARM32, x86, and x86_64 architectures
4. Optimized APK under 150MB with modular library downloads
5. **Enhanced User Experience**
6. Intuitive interface with minimal learning curve
7. Responsive design adaptable to various screen sizes
8. Accessibility features for inclusive design
9. **Robust Performance**
10. Fast code execution and compilation
11. Efficient memory management
12. Battery-optimized background processing

Educational Deliverables:

1. **Comprehensive Documentation**
2. User manual and developer guide
3. API documentation for extensibility
4. Video tutorials and getting started guide
5. **Sample Projects and Templates**
6. Beginner to advanced Python projects
7. Programming exercises and challenges
8. Real-world application examples

7. Innovation and Unique Features

Key Innovations:

1. **AI-Powered Code Assistant**: Intelligent code suggestions and error fixing
2. **Mobile-Optimized Interface**: Touch-friendly coding experience
3. **Integrated Learning Platform**: Built-in tutorials and skill assessments
4. **Cloud-Native Architecture**: Seamless project synchronization
5. **Community Features**: Code sharing and collaboration tools

Competitive Advantages:

- **Superior Performance**: Optimized Python runtime for mobile
- **Modern Design**: Latest Material Design 3 implementation
- **Educational Focus**: Built-in learning resources and tutorials
- **Extensibility**: Plugin architecture for future enhancements
- **Open Source**: Community-driven development and contributions

8. Technology Stack

Frontend:

- **Language**: Kotlin
- **UI Framework**: Jetpack Compose
- **Design System**: Material Design 3
- **Navigation**: Compose Navigation

Backend/Runtime:

- **Python Version**: Python 3.12+
- **Native Libraries**: Cross-compiled Python binaries
- **Package Management**: Pip integration
- **Terminal Emulation**: Enhanced Termux libraries

Development Tools:

- **IDE**: Android Studio (Latest)
- **Version Control**: Git with GitHub integration
- **CI/CD**: GitHub Actions
- **Testing**: JUnit, Espresso, Compose Testing

Third-Party Libraries:

- **Code Editor**: Enhanced Sora Editor (Rosemoe)
- **Terminal**: Termux components
- **Networking**: Retrofit, OkHttp
- **Database**: Room for local storage
- **Analytics**: Firebase Analytics

9. Feasibility Analysis

Technical Feasibility:

- **High**: Based on existing KtxPy foundation
- Proven cross-compilation of Python for Android
- Established libraries and frameworks available
- Strong community support for open-source components

Economic Feasibility:

- **High**: Free and open-source development tools
- No licensing costs for core technologies
- Potential for monetization through premium features
- Low infrastructure costs with cloud services

Time Feasibility:

- **Medium-High**: 6-month timeline is adequate
- Iterative development approach
- Parallel development of core features
- Sufficient buffer time for testing and optimization

10. Risk Assessment

Technical Risks:

- **Medium**: Cross-platform compatibility issues
- **Low**: Performance optimization challenges
- **Medium**: Python library compatibility

Mitigation Strategies:

- Extensive testing on multiple device configurations
- Continuous integration and automated testing
- Gradual feature rollout with user feedback
- Active community engagement for issue resolution

11. Project Impact

Academic Impact:

- Demonstrates advanced mobile application development skills
- Shows proficiency in multiple programming languages and frameworks
- Exhibits understanding of user experience design principles
- Provides practical solution to real-world problems

Social Impact:

- Democratizes programming education
- Enables coding in resource-constrained environments
- Supports mobile-first learning approaches
- Contributes to open-source community

Professional Impact:

- Portfolio project demonstrating technical expertise
- Experience with modern Android development practices
- Understanding of IDE development complexities
- Community engagement and project management skills

12. Success Metrics

Quantitative Metrics:

- **Performance**: App startup time under 3 seconds
- **Stability**: 99.5% crash-free rate
- **User Engagement**: Average session time > 15 minutes

- **Download Metrics**: Target 10,000+ downloads in first 6 months

Qualitative Metrics:

- **User Satisfaction**: 4.5+ rating on Google Play Store
- **Code Quality**: Maintainable and well-documented codebase
- **Community Adoption**: Active GitHub repository with contributions
- **Educational Value**: Positive feedback from educational institutions

Conclusion

Python Pocket IDE represents a significant advancement in mobile development environments, combining the power of desktop IDEs with the convenience of mobile accessibility. This project will not only fulfill the academic requirements of the final year project but also contribute meaningfully to the programming community by providing a high-quality, open-source development tool.

The project leverages cutting-edge Android development technologies while addressing real-world needs in the programming education and mobile development domains. Through careful planning, iterative development, and community engagement, PPIDE aims to set new standards for mobile programming environments.

Prepared by: Ankit
Date: January 2025
Supervisor: [To be assigned]
Department: Computer Science and Engineering
Institution: Chandigarh University

Software Development Life Cycle (SDLC) Plan

Python Pocket IDE (PPIDE) Final Year Project

Project Information

- **Project**: Python Pocket IDE (PPIDE)
- **Student**: Ankit
- **Course**: MCA Final Year (Batch Jul 2023)
- **University**: Chandigarh University
- **Duration**: 6 Months (January 2025 - June 2025)
- **SDLC Model**: Agile with Iterative Development

---

1. SDLC Model Selection

Chosen Model: Agile with Iterative Development

Rationale: - **Flexibility**: Allows for changes based on user feedback and technical discoveries - **Risk Management**: Early identification and mitigation of technical challenges - **Incremental Delivery**: Functional features delivered in each sprint - **Educational Value**: Provides hands-on experience with industry-standard practices - **Time Constraints**: 6-month timeline requires efficient development cycles

Development Approach:

- **Sprint Duration**: 2 weeks
- **Total Sprints**: 12 sprints
- **Release Cycles**: 3 major releases (Alpha, Beta, Final)
- **Review Cycles**: Weekly progress reviews, bi-weekly sprint reviews

---

2. Phase-wise Development Plan

Phase 1: Project Initiation & Planning (January 2025)

Duration: 2 weeks (Sprint 1)

Activities:

1. **Requirement Analysis**
2. Detailed feature specification
3. User story creation
4. Acceptance criteria definition
5. Technical requirement documentation
6. **System Design**
7. Architecture design
8. Database schema design
9. UI/UX wireframes
10. API design documentation
11. **Project Setup**
12. Development environment setup
13. Version control initialization
14. CI/CD pipeline configuration
15. Team collaboration tools setup

Deliverables:

- [ ] Software Requirement Specification (SRS)
- [ ] System Design Document (SDD)
- [ ] Project Plan with timelines
- [ ] Risk Assessment Document
- [ ] Development environment setup

Success Criteria:

- All requirements documented and approved
- Development environment fully configured
- Project timeline agreed upon
- Risk mitigation strategies defined

Phase 2: Core Foundation Development (January-February 2025)

Duration: 4 weeks (Sprint 2-3)

Sprint 2 (Week 3-4): Basic Infrastructure

Focus: Core Android app structure and Python integration

Activities:

1. **Android App Foundation**
2. Basic app structure with Jetpack Compose
3. Material Design 3 implementation
4. Navigation system setup
5. Theme and styling framework
6. **Python Runtime Integration**
7. Python environment extraction and setup
8. Basic terminal emulation
9. File system integration
10. Permission handling

Deliverables:

- [ ] Basic Android app with navigation
- [ ] Python runtime successfully integrated
- [ ] Basic file operations working
- [ ] Initial UI framework implemented

Sprint 3 (Week 5-6): Basic Editor Implementation

Focus: Core code editor functionality

Activities:

1. **Code Editor Integration**
2. Sora Editor integration and customization
3. Basic syntax highlighting for Python
4. File creation, opening, and saving
5. Basic text editing operations
6. **Basic Python Execution**
7. Simple Python script execution
8. Output display in terminal
9. Error handling and display
10. Basic debugging information

Deliverables:

- [ ] Functional code editor with syntax highlighting
- [ ] Python script execution capability
- [ ] Basic file management system
- [ ] Terminal output display

Success Criteria for Phase 2:

- Users can create and edit Python files
- Basic Python scripts can be executed
- App follows Material Design 3 guidelines
- Core functionality stable and tested

Phase 3: Advanced Features Development (February-April 2025)

Duration: 8 weeks (Sprint 4-7)

Sprint 4 (Week 7-8): Enhanced Editor Features

Focus: Advanced editing capabilities

Activities:

1. **Advanced Editor Features**
2. Code completion and IntelliSense
3. Find and replace functionality
4. Code folding and line numbers
5. Multiple file tabs
6. **Improved Python Integration**
7. Enhanced error detection and highlighting
8. Pip package management
9. Python library support
10. Interactive Python shell

Deliverables:

- [ ] Code completion working
- [ ] Multi-tab editing support
- [ ] Pip integration functional
- [ ] Enhanced error reporting

Sprint 5 (Week 9-10): Project Management Features

Focus: File and project organization

Activities:

1. **Project Management**
2. Project creation and organization
3. File explorer with tree view
4. Import/export functionality
5. Project templates
6. **Enhanced UI/UX**
7. Improved navigation drawer
8. Settings and preferences screen
9. Theme customization options
10. Responsive design improvements

Deliverables:

- [ ] Project management system
- [ ] Enhanced file explorer
- [ ] Settings and customization options
- [ ] Improved overall UI/UX

Sprint 6 (Week 11-12): Debugging and Tools

Focus: Development tools and debugging

Activities:

1. **Debugging Features**
2. Breakpoint support
3. Step-through debugging
4. Variable inspection
5. Call stack visualization
6. **Development Tools**
7. Code formatting and linting
8. Documentation generator
9. Performance profiling
10. Memory usage monitoring

Deliverables:

- [ ] Basic debugging functionality
- [ ] Code formatting tools
- [ ] Performance monitoring
- [ ] Documentation generation

Sprint 7 (Week 13-14): Educational Content

Focus: Learning and tutorial features

Activities:

1. **Educational Modules**
2. Interactive Python tutorials
3. Code examples and templates
4. Programming exercises
5. Skill assessment tools
6. **Sample Projects**
7. Beginner to advanced Python projects
8. Game development examples
9. Data science templates
10. Web scraping examples

Deliverables:

- [ ] Interactive tutorial system
- [ ] Comprehensive sample projects

- [ ] Educational content management
- [ ] Progress tracking system

Success Criteria for Phase 3:

- Advanced editing features fully functional
- Debugging tools operational
- Educational content accessible and engaging
- All features tested and optimized

Phase 4: Integration & Enhancement (April-May 2025)

Duration: 4 weeks (Sprint 8-9)

Sprint 8 (Week 15-16): Cloud Integration

Focus: Cloud features and synchronization

Activities:

1. **Cloud Storage Integration**
2. Google Drive integration
3. Project backup and sync
4. Cross-device compatibility
5. Offline mode support
6. **Sharing and Collaboration**
7. Code sharing functionality
8. Export to various formats
9. Social media integration
10. Community features

Deliverables:

- [ ] Cloud storage integration
- [ ] Project synchronization
- [ ] Sharing capabilities
- [ ] Community features

Sprint 9 (Week 17-18): Performance Optimization

Focus: Performance and stability improvements

Activities:

1. **Performance Optimization**
2. Memory usage optimization
3. Battery life improvements
4. App startup time reduction
5. Code execution efficiency
6. **Stability Improvements**
7. Crash prevention and handling
8. Error recovery mechanisms
9. Data persistence improvements
10. Background task optimization

Deliverables:

- [ ] Optimized app performance
- [ ] Enhanced stability
- [ ] Improved user experience
- [ ] Battery life optimization

Success Criteria for Phase 4:

- Cloud features working seamlessly
- App performance meets target metrics
- Stability issues resolved
- User experience polished

Phase 5: Testing & Quality Assurance (May 2025)

Duration: 3 weeks (Sprint 10-11)

Sprint 10 (Week 19-20): Comprehensive Testing

Focus: Testing and bug fixing

Activities:

1. **Automated Testing**
2. Unit test development
3. Integration test implementation
4. UI test automation
5. Performance test execution
6. **Manual Testing**
7. User acceptance testing
8. Cross-device compatibility testing
9. Edge case testing
10. Accessibility testing

Deliverables:

- [ ] Comprehensive test suite
- [ ] Bug reports and fixes
- [ ] Performance benchmarks
- [ ] Accessibility compliance

Sprint 11 (Week 21): Bug Fixing & Polish

Focus: Final bug fixes and UI polish

Activities:

1. **Bug Resolution**
2. Critical bug fixes
3. Performance issue resolution
4. UI/UX improvements
5. Documentation updates
6. **Final Polish**
7. Icon and branding finalization
8. Animation and transition improvements
9. Help and tutorial refinements
10. App store preparation

Deliverables:

- [ ] All critical bugs resolved
- [ ] Final UI/UX polish
- [ ] Complete documentation
- [ ] App store ready build

Success Criteria for Phase 5:

- All critical bugs resolved
- App meets quality standards
- Documentation complete
- Ready for release

Phase 6: Deployment & Documentation (June 2025)

Duration: 2 weeks (Sprint 12)

Sprint 12 (Week 22-23): Final Deployment

Focus: Release preparation and documentation

Activities:

1. **Release Preparation**
2. Final build optimization
3. App store submission preparation
4. Release notes creation
5. Distribution strategy planning
6. **Project Documentation**
7. Final project report
8. Technical documentation
9. User manual creation
10. Video demonstration

Deliverables:

- [ ] Production-ready application

- [ ] App store submission
- [ ] Complete project documentation
- [ ] Project presentation materials

Success Criteria for Phase 6:

- Application successfully deployed
- All documentation completed
- Project presentation ready
- Academic requirements fulfilled

---

3. Risk Management Plan

Technical Risks

Risk	Probability	Impact	Mitigation Strategy
Python integration complexity	Medium	High	Early prototyping, community support
Performance issues on low-end devices	High	Medium	Continuous performance testing
Android compatibility issues	Medium	Medium	Testing on multiple devices
Third-party library dependencies	Low	High	Alternative library research

Project Risks

Risk	Probability	Impact	Mitigation Strategy
Timeline delays	Medium	High	Buffer time allocation, scope adjustment
Feature scope creep	High	Medium	Strict change control process
Technical skill gaps	Low	Medium	Continuous learning, mentor support
External dependency failures	Low	High	Backup solutions, fallback plans

---

4. Quality Assurance Plan

Code Quality Standards

- **Code Coverage**: Minimum 80% test coverage
- **Code Review**: All code peer-reviewed before merge
- **Documentation**: Comprehensive inline documentation
- **Coding Standards**: Kotlin coding conventions followed

Testing Strategy

1. **Unit Testing**: All business logic components
2. **Integration Testing**: API and database interactions
3. **UI Testing**: User interface functionality
4. **Performance Testing**: Memory, battery, and speed
5. **Compatibility Testing**: Multiple Android versions and devices

Quality Metrics

- **Crash Rate**: < 0.5% (Target: 0.1%)
- **ANR Rate**: < 0.1%
- **App Start Time**: < 3 seconds
- **Memory Usage**: < 200MB average
- **Battery Drain**: Minimal background usage

---

5. Communication Plan

Stakeholder Communication

- **Weekly Reports**: Progress updates to supervisor
- **Bi-weekly Reviews**: Sprint demonstrations
- **Monthly Presentations**: Major milestone reviews
- **Final Presentation**: Complete project demonstration

Documentation Schedule

- **Weekly**: Sprint reports and progress updates
- **Bi-weekly**: Technical documentation updates
- **Monthly**: Comprehensive progress reports
- **Final**: Complete project documentation

---

6. Tools and Technologies

Development Tools

- **IDE**: Android Studio (Latest)
- **Version Control**: Git with GitHub
- **Project Management**: GitHub Projects, Trello
- **Communication**: Slack, Email
- **Documentation**: Markdown, GitHub Wiki

Testing Tools

- **Unit Testing**: JUnit, Mockito
- **UI Testing**: Espresso, Compose Testing
- **Performance**: Android Profiler
- **Static Analysis**: SonarQube, Android Lint

CI/CD Pipeline

- **Build Automation**: GitHub Actions
- **Testing**: Automated test execution
- **Code Quality**: Automated code analysis
- **Deployment**: Automated APK generation

---

7. Success Criteria

Academic Success Criteria

- [ ] Project completed within timeline
- [ ] All deliverables submitted on time
- [ ] Technical requirements fulfilled
- [ ] Documentation standards met
- [ ] Successful project presentation

Technical Success Criteria

- [ ] Functional Python IDE for Android
- [ ] Performance targets achieved
- [ ] Quality metrics satisfied
- [ ] User acceptance criteria met
- [ ] Open-source contribution ready

Learning Objectives Met

- [ ] Advanced Android development skills
- [ ] Project management experience
- [ ] Software engineering best practices
- [ ] Problem-solving and innovation
- [ ] Professional documentation skills

Document Prepared by: Ankit
Date: January 2025
Last Updated: January 2025
Version: 1.0

Software Requirements Specification (SRS)

Python Pocket IDE (PPIDE)

Document Information

- **Document Title**: Software Requirements Specification
- **Project**: Python Pocket IDE (PPIDE)
- **Version**: 1.0
- **Date**: January 2025
- **Prepared by**: Ankit
- **Approved by**: [Supervisor Name]

Table of Contents

---

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for the Python Pocket IDE (PPIDE) mobile application. The document is intended for developers, testers, project managers, and stakeholders involved in the development and deployment of the application.

1.2 Document Scope

This SRS covers the complete functionality of Python Pocket IDE, including: - Core IDE features for Python development - User interface and user experience requirements - Performance and security requirements - Integration requirements with external services - Platform-specific requirements for Android

1.3 Definitions, Acronyms, and Abbreviations

- **PPIDE**: Python Pocket IDE
- **IDE**: Integrated Development Environment
- **API**: Application Programming Interface
- **UI**: User Interface
- **UX**: User Experience
- **SDK**: Software Development Kit
- **APK**: Android Package
- **CI/CD**: Continuous Integration/Continuous Deployment

1.4 References

- Android Developer Documentation
- Material Design 3 Guidelines
- Python Official Documentation
- IEEE Std 830-1998 (SRS Standard)

1.5 Overview

The following sections provide a detailed description of the software requirements, including functional and non-functional requirements, external interfaces, and system constraints.

---

2. Overall Description

2.1 Product Perspective

Python Pocket IDE is a standalone mobile application designed for Android devices. It provides a comprehensive Python development environment that runs entirely on mobile devices without requiring external servers for core functionality.

2.1.1 System Interfaces

- **Operating System**: Android 8.0 (API level 26) and above
- **Hardware**: ARM, ARM64, x86, x86_64 architectures
- **Storage**: Local file system with cloud backup integration
- **Network**: Internet connectivity for package downloads and cloud features

2.1.2 User Interfaces

- Modern Material Design 3 interface
- Touch-optimized controls and gestures
- Responsive design for tablets and phones
- Dark and light theme support

2.1.3 Hardware Interfaces

- **Touch Screen**: Primary input method
- **Keyboard**: Virtual and physical keyboard support
- **Storage**: Internal and external storage access
- **Network**: Wi-Fi and mobile data connectivity

2.2 Product Functions

The main functions of Python Pocket IDE include:

1. **Code Editing**: Advanced text editor with Python syntax highlighting
2. **Code Execution**: Python script execution and output display
3. **Project Management**: File and project organization
4. **Package Management**: Python package installation and management
5. **Debugging**: Basic debugging tools and error reporting
6. **Learning**: Educational content and tutorials
7. **Sharing**: Code sharing and export capabilities

2.3 User Classes and Characteristics

2.3.1 Primary Users

- **Students**: Learning Python programming (Beginner to Intermediate)
- **Educators**: Teaching Python in academic settings
- **Mobile Developers**: Quick prototyping and testing
- **Hobbyist Programmers**: Casual programming on mobile devices

2.3.2 Secondary Users

- **Professional Developers**: Emergency coding and quick fixes
- **Interview Candidates**: Practice coding problems
- **Content Creators**: Creating programming tutorials

2.4 Operating Environment

- **Client Platform**: Android 8.0+ (API level 26+)
- **Memory**: Minimum 3GB RAM (Recommended: 4GB+)
- **Storage**: Minimum 2GB free space
- **Network**: Optional for cloud features and package downloads
- **Screen**: 5.0" minimum (Optimized for 6.0"+)

2.5 Design and Implementation Constraints

- **Platform**: Android-only in initial version
- **Programming Language**: Kotlin for Android development
- **UI Framework**: Jetpack Compose
- **Python Version**: Python 3.12+
- **Architecture**: Clean Architecture with MVVM pattern
- **Minimum SDK**: Android 8.0 (API 26)
- **Target SDK**: Latest Android version

2.6 Assumptions and Dependencies

- Users have basic familiarity with mobile applications
- Devices have sufficient storage and memory
- Internet connectivity available for optional features
- Google Play Store availability for distribution

---

3. System Features

3.1 Code Editor Module

3.1.1 Feature Description

A comprehensive code editor specifically designed for Python development on mobile devices.

3.1.2 Functional Requirements

FR-CE-001: Syntax Highlighting - The system shall provide Python syntax highlighting - The system shall support multiple color themes - The system shall highlight syntax errors in real-time

FR-CE-002: Code Completion - The system shall provide intelligent code completion - The system shall suggest

Python keywords, functions, and variables - The system shall support context-aware suggestions

FR-CE-003: Text Editing Operations - The system shall support standard text operations (cut, copy, paste) - The system shall provide undo/redo functionality - The system shall support find and replace operations

FR-CE-004: File Management - The system shall allow creating new Python files - The system shall support opening and saving files - The system shall provide recent files list

FR-CE-005: Multi-tab Support - The system shall support multiple file tabs - The system shall indicate unsaved changes - The system shall allow switching between open files

3.2 Python Runtime Module

3.2.1 Feature Description

Integrated Python runtime environment for executing Python code directly on Android devices.

3.2.2 Functional Requirements

FR-PR-001: Python Execution - The system shall execute Python scripts - The system shall display execution output - The system shall show execution time and resource usage

FR-PR-002: Interactive Shell - The system shall provide an interactive Python shell (REPL) - The system shall support multi-line input - The system shall maintain session history

FR-PR-003: Error Handling - The system shall display Python runtime errors - The system shall highlight error lines in the editor - The system shall provide detailed error messages

FR-PR-004: Package Management - The system shall support pip package installation - The system shall list installed packages - The system shall allow package uninstallation

3.3 Project Management Module

3.3.1 Feature Description

Comprehensive project and file management system for organizing Python projects.

3.3.2 Functional Requirements

FR-PM-001: Project Creation - The system shall allow creating new projects - The system shall provide project templates - The system shall organize files in project structure

FR-PM-002: File Explorer - The system shall display project files in tree view - The system shall support file operations (create, delete, rename) - The system shall show file types with appropriate icons

FR-PM-003: Import/Export - The system shall import projects from zip files - The system shall export projects to various formats - The system shall support individual file import/export

3.4 Educational Module

3.4.1 Feature Description

Integrated learning platform with tutorials, examples, and exercises for Python programming.

3.4.2 Functional Requirements

FR-ED-001: Interactive Tutorials - The system shall provide step-by-step Python tutorials - The system shall track learning progress - The system shall include hands-on coding exercises

FR-ED-002: Sample Projects - The system shall include beginner to advanced sample projects - The system shall provide project explanations - The system shall allow modification of sample code

FR-ED-003: Code Templates - The system shall provide code templates for common patterns - The system shall support custom template creation - The system shall categorize templates by difficulty level

3.5 User Interface Module

3.5.1 Feature Description

Modern, intuitive user interface following Material Design 3 principles.

3.5.2 Functional Requirements

FR-UI-001: Navigation - The system shall provide intuitive navigation between features - The system shall include a navigation drawer for main sections - The system shall support gesture-based navigation

FR-UI-002: Theming - The system shall support dark and light themes - The system shall provide multiple editor color schemes - The system shall remember user theme preferences

FR-UI-003: Settings - The system shall provide comprehensive settings screen - The system shall allow customization of editor preferences - The system shall support app-wide configuration options

3.6 Cloud Integration Module

3.6.1 Feature Description

Optional cloud features for project backup, synchronization, and sharing.

3.6.2 Functional Requirements

FR-CI-001: Project Backup - The system shall backup projects to cloud storage - The system shall support automatic backup scheduling - The system shall allow manual backup initiation

FR-CI-002: Synchronization - The system shall sync projects across devices - The system shall handle synchronization conflicts - The system shall work offline with sync when connected

FR-CI-003: Sharing - The system shall allow sharing code via various methods - The system shall support exporting to GitHub - The system shall generate shareable links for projects

---

4. External Interface Requirements

4.1 User Interfaces

4.1.1 Main Interface Components

1. **Home Screen**: Project listing and navigation
2. **Editor Screen**: Code editing interface
3. **Terminal Screen**: Python execution and output
4. **File Manager**: Project and file management
5. **Settings Screen**: App configuration
6. **Tutorial Screen**: Educational content

4.1.2 Interface Standards

- Material Design 3 compliance
- Accessibility support (TalkBack, high contrast)
- Responsive design for different screen sizes
- Touch-friendly controls with appropriate sizing

4.2 Hardware Interfaces

4.2.1 Input Devices

- **Touch Screen**: Primary input for all interactions
- **Virtual Keyboard**: Text input with code-friendly layout
- **Physical Keyboard**: Optional external keyboard support
- **Voice Input**: For accessibility and hands-free operation

4.2.2 Storage Interfaces

- **Internal Storage**: App data and user projects
- **External Storage**: Project import/export
- **Cloud Storage**: Backup and synchronization

4.3 Software Interfaces

4.3.1 Operating System

- **Android API**: Version 26 (Android 8.0) minimum
- **File System**: Standard Android file operations
- **Permissions**: Storage, network access as needed

4.3.2 External Libraries

- **Python Runtime**: Embedded Python 3.12+
- **Code Editor**: Enhanced Sora Editor library
- **Terminal Emulation**: Termux terminal components
- **Cloud APIs**: Google Drive, GitHub APIs

4.4 Communication Interfaces

4.4.1 Network Protocols

- **HTTPS**: Secure web communication
- **REST APIs**: Cloud service integration
- **WebSocket**: Real-time features (future)

4.4.2 Data Formats

- **JSON**: Configuration and data exchange
- **XML**: Android resource files
- **ZIP**: Project packaging and templates

---

5. System Requirements

5.1 Hardware Requirements

5.1.1 Minimum Requirements

- **Processor**: ARM Cortex-A53 or equivalent
- **RAM**: 3GB minimum
- **Storage**: 2GB free space
- **Screen**: 5.0" with 720p resolution
- **Network**: Wi-Fi or mobile data (optional)

5.1.2 Recommended Requirements

- **Processor**: ARM Cortex-A75 or equivalent
- **RAM**: 4GB or more
- **Storage**: 4GB free space
- **Screen**: 6.0" with 1080p resolution
- **Network**: High-speed Wi-Fi or 4G/5G

5.2 Software Requirements

5.2.1 Platform Requirements

- **Operating System**: Android 8.0 (API 26) minimum
- **Target OS**: Android 14 (API 34)
- **Architecture Support**: ARM64, ARM32, x86, x86_64

5.2.2 Development Requirements

- **Build System**: Android Gradle Plugin 8.0+
- **Compile SDK**: Android 34
- **Java Version**: Java 11 or higher
- **Kotlin Version**: 1.9.0 or higher

---

6. Non-Functional Requirements

6.1 Performance Requirements

6.1.1 Response Time

- **App Launch**: Under 3 seconds on recommended hardware
- **File Opening**: Under 2 seconds for files up to 1MB
- **Code Execution**: Comparable to desktop Python performance
- **UI Responsiveness**: 60 FPS during normal operation

6.1.2 Throughput

- **File Operations**: Handle files up to 10MB efficiently
- **Concurrent Operations**: Support multiple background tasks
- **Memory Usage**: Under 200MB average, 400MB peak

6.1.3 Capacity

- **Projects**: Support 100+ projects per device
- **Files per Project**: Up to 1000 files
- **File Size**: Individual files up to 50MB
- **Total Storage**: Up to 10GB of user data

6.2 Safety and Security Requirements

6.2.1 Data Security

- **Local Data**: Encrypted storage for sensitive information
- **Cloud Data**: Secure transmission and storage
- **User Privacy**: No unauthorized data collection
- **Permissions**: Minimal required permissions

6.2.2 Code Security

- **Sandbox**: Python execution in isolated environment
- **File Access**: Restricted to app directories
- **Network Access**: Controlled network operations
- **Package Security**: Verified package downloads

6.3 Software Quality Attributes

6.3.1 Reliability

- **Availability**: 99.9% uptime for core features
- **Fault Tolerance**: Graceful handling of errors
- **Recovery**: Automatic recovery from crashes

- **Data Integrity**: Protection against data loss

6.3.2 Usability

- **Learnability**: Intuitive interface for new users
- **Efficiency**: Productive workflow for experienced users
- **Memorability**: Consistent interface patterns
- **Error Prevention**: Clear feedback and validation

6.3.3 Maintainability

- **Modularity**: Clean separation of concerns
- **Documentation**: Comprehensive code documentation
- **Testability**: High test coverage and automation
- **Extensibility**: Plugin architecture for future features

6.3.4 Portability

- **Platform Independence**: Android architecture support
- **Version Compatibility**: Android 8.0+ support
- **Device Adaptation**: Responsive design for all screen sizes
- **Backward Compatibility**: Maintain compatibility with older versions

---

7. Other Requirements

7.1 Legal Requirements

- **Open Source License**: GPL-3.0 license compliance
- **Third-party Licenses**: Proper attribution and compliance
- **Privacy Policy**: GDPR and local privacy law compliance
- **Terms of Service**: Clear user agreement

7.2 Internationalization

- **Languages**: English primary, with framework for localization
- **Cultural Adaptation**: Appropriate for international users
- **Text Direction**: Support for LTR languages initially
- **Number Formats**: Locale-appropriate formatting

7.3 Documentation Requirements

- **User Manual**: Comprehensive user guide
- **Developer Documentation**: Technical implementation details
- **API Documentation**: For future extensibility
- **Release Notes**: Version-specific changes and updates

7.4 Training Requirements

- **User Training**: Interactive tutorials and help system
- **Video Tutorials**: Getting started and advanced features
- **Sample Projects**: Hands-on learning examples
- **Community Support**: Forums and knowledge base

Appendices

Appendix A: User Story Examples

US-001: As a student, I want to write and run Python code on my phone so that I can practice programming anywhere.

US-002: As an educator, I want to share coding exercises with students so that they can learn interactively.

US-003: As a developer, I want to quickly test Python code snippets so that I can verify algorithms on the go.

Appendix B: Use Case Diagrams

[Use case diagrams would be included here in the actual document]

Appendix C: Data Flow Diagrams

[Data flow diagrams would be included here in the actual document]

Appendix D: Mockups and Wireframes

[UI mockups and wireframes would be included here in the actual document]

Document Control - **Version**: 1.0 - **Last Modified**: January 2025 - **Next Review**: February 2025 - **Approval Status**: Draft

Implementation Plan - Python Pocket IDE (PPIDE)

Enhanced Features and Architecture

Project Information

- **Project**: Python Pocket IDE (PPIDE)
- **Student**: Ankit
- **Course**: MCA Final Year (Batch Jul 2023)
- **University**: Chandigarh University
- **Duration**: 6 Months (January 2025 - June 2025)

---

1. Enhanced Features Overview

1.1 Core Enhancements Over KtxPy

Advanced Code Editor Features

- **AI-Powered Code Completion**: Intelligent suggestions using local AI models
- **Advanced Syntax Analysis**: Real-time error detection and suggestions
- **Code Refactoring Tools**: Automated code improvement suggestions
- **Advanced Find & Replace**: Regex support and project-wide search
- **Code Folding**: Collapsible code blocks for better navigation
- **Minimap**: Code overview for large files
- **Split View**: Side-by-side file comparison

Enhanced Python Environment

- **Python 3.12+ Runtime**: Latest Python version with performance improvements
- **Package Manager UI**: Graphical interface for pip operations
- **Virtual Environment Support**: Isolated Python environments per project
- **Interactive Debugger**: Breakpoints, step-through debugging, variable inspection
- **Performance Profiler**: Code execution analysis and optimization suggestions
- **Memory Monitor**: Real-time memory usage tracking

Advanced Project Management

- **Git Integration**: Version control with commit, push, pull operations
- **Project Templates**: Pre-configured project structures
- **Build System**: Custom build configurations and scripts
- **Dependency Management**: Automatic dependency resolution
- **Project Analytics**: Code metrics and quality analysis
- **Export Options**: Multiple format exports (GitHub, GitLab, ZIP)

Educational Platform

- **Interactive Tutorials**: Step-by-step guided learning
- **Skill Assessment**: Programming challenges and evaluations
- **Progress Tracking**: Learning path monitoring
- **Code Challenges**: Daily programming problems
- **Community Features**: Code sharing and collaboration
- **Certification System**: Achievement badges and certificates

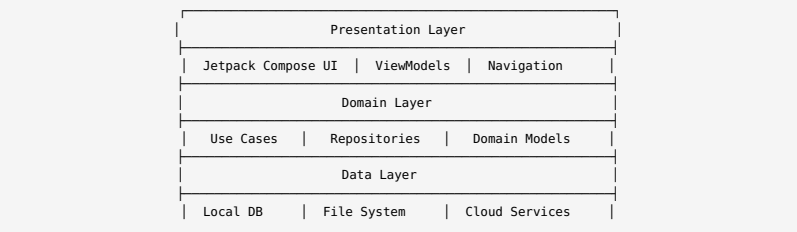
Modern UI/UX

- **Material Design 3**: Latest design system implementation
- **Dynamic Theming**: Adaptive color schemes
- **Accessibility Features**: Screen reader support, high contrast modes
- **Gesture Navigation**: Touch-friendly controls
- **Tablet Optimization**: Responsive design for larger screens
- **Customizable Workspace**: Personalized layout options

---

2. Technical Architecture Enhancement

2.1 Architecture Pattern: MVVM + Clean Architecture



```
│  (Room)       │  Operations      │  (Firebase/Drive)   │
└──────────────────────────────────────────────────────┘
```

2.2 Module Structure

Core Modules

- **app**: Main application module
- **core-ui**: Shared UI components and themes
- **core-data**: Data layer implementations
- **core-domain**: Business logic and use cases
- **feature-editor**: Code editing functionality
- **feature-terminal**: Python execution environment
- **feature-projects**: Project management
- **feature-learning**: Educational content
- **feature-settings**: App configuration

New Enhanced Modules

- **feature-ai**: AI-powered code assistance
- **feature-git**: Version control integration
- **feature-debugger**: Advanced debugging tools
- **feature-analytics**: Code analysis and metrics
- **feature-cloud**: Cloud synchronization
- **feature-community**: Social features

---

3. Development Roadmap

Phase 1: Foundation Enhancement (Weeks 1-4)

Week 1-2: Project Setup and Architecture

- [x] Project structure reorganization
- [ ] Dependency injection setup (Hilt)
- [ ] Database schema design (Room)
- [ ] Navigation graph enhancement
- [ ] CI/CD pipeline setup

Week 3-4: Core UI Framework

- [ ] Material Design 3 implementation
- [ ] Custom theme system
- [ ] Responsive layout components
- [ ] Accessibility framework
- [ ] Animation system

Phase 2: Enhanced Editor (Weeks 5-8)

Week 5-6: Advanced Editor Features

- [ ] Sora Editor customization and enhancement
- [ ] Advanced syntax highlighting
- [ ] Code completion engine
- [ ] Error detection system
- [ ] Code folding implementation

Week 7-8: AI Integration

- [ ] Local AI model integration
- [ ] Intelligent code suggestions
- [ ] Code refactoring suggestions
- [ ] Documentation generation
- [ ] Code quality analysis

Phase 3: Python Environment Enhancement (Weeks 9-12)

Week 9-10: Advanced Python Runtime

- [ ] Python 3.12 integration
- [ ] Virtual environment support
- [ ] Enhanced package management
- [ ] Interactive shell improvements
- [ ] Performance optimization

Week 11-12: Debugging and Profiling

- [ ] Breakpoint system implementation
- [ ] Variable inspection interface
- [ ] Call stack visualization

- [ ] Performance profiler
- [ ] Memory monitoring tools

Phase 4: Project Management (Weeks 13-16)

Week 13-14: Git Integration

- [ ] Git library integration
- [ ] Repository management UI
- [ ] Commit and push functionality
- [ ] Branch management
- [ ] Merge conflict resolution

Week 15-16: Advanced Project Features

- [ ] Project templates system
- [ ] Build configuration
- [ ] Dependency management
- [ ] Export functionality
- [ ] Project analytics

Phase 5: Educational Platform (Weeks 17-20)

Week 17-18: Learning Content

- [ ] Interactive tutorial engine
- [ ] Content management system
- [ ] Progress tracking database
- [ ] Achievement system
- [ ] Skill assessment tools

Week 19-20: Community Features

- [ ] Code sharing platform
- [ ] User profiles and portfolios
- [ ] Discussion forums
- [ ] Code review system
- [ ] Collaboration tools

Phase 6: Cloud and Integration (Weeks 21-22)

Week 21: Cloud Services

- [ ] Firebase integration
- [ ] Google Drive sync
- [ ] GitHub API integration
- [ ] Cloud backup system
- [ ] Cross-device synchronization

Week 22: Final Polish

- [ ] Performance optimization
- [ ] Bug fixes and stability
- [ ] UI/UX refinements
- [ ] Documentation completion
- [ ] Release preparation

---

4. Technical Implementation Details

4.1 Enhanced Code Editor Implementation

Advanced Syntax Highlighting

```
                class EnhancedPythonLanguage : Language {
override fun analyze(content: ContentReference): AnalyzeManager.State {
                    return PythonAnalyzer().analyze(content)
                                }

                fun getAdvancedHighlighting(): ColorScheme {
                        return PythonColorScheme().apply {
            // Enhanced color schemes for different Python constructs
        setColor(EditorColorScheme.KEYWORD, Color.parseColor("#FF5722"))
        setColor(EditorColorScheme.STRING, Color.parseColor("#4CAF50"))
                    // ... more advanced coloring
                                }
                            }
                        }
```

AI-Powered Code Completion

```
class AICodeCompletionProvider : CompletionPublisher {
```

```
    private val aiModel = LocalAIModel()

    override fun requireAutoComplete(
        content: ContentReference,
        position: CharPosition,
        publisher: CompletionPublisher
    ) {
        val context = extractCodeContext(content, position)
        val suggestions = aiModel.generateSuggestions(context)
        publisher.addItems(suggestions.map { it.toCompletionItem() })
    }
}
```

4.2 Enhanced Python Runtime

Virtual Environment Management

```
class VirtualEnvironmentManager {
    fun createVirtualEnv(projectPath: String, pythonVersion: String) {
        val venvPath = "$projectPath/.venv"
        executeCommand("python$pythonVersion -m venv $venvPath")
        updateProjectConfig(projectPath, venvPath)
    }

    fun activateVirtualEnv(projectPath: String) {
        val config = loadProjectConfig(projectPath)
        val activationScript = "${config.venvPath}/bin/activate"
        currentEnvironment = VirtualEnvironment(activationScript)
    }
}
```

Advanced Debugging System

```
class PythonDebugger {
    private val breakpoints = mutableMapOf<String, Set<Int>>()

    fun setBreakpoint(file: String, line: Int) {
        breakpoints.getOrPut(file) { mutableSetOf() }.add(line)
        updateDebugSession()
    }

    fun startDebugging(script: String) {
        val debugProcess = PythonDebugProcess(script, breakpoints)
        debugProcess.onBreakpointHit { file, line, variables ->
            showDebugInterface(file, line, variables)
        }
    }
}
```

4.3 Git Integration

Repository Management

```
class GitRepository(private val projectPath: String) {
    private val git = Git.open(File(projectPath))

    fun commit(message: String, files: List<String>) {
        files.forEach { git.add().addFilepattern(it).call() }
        git.commit().setMessage(message).call()
    }

    fun push(remote: String = "origin", branch: String = "main") {
        git.push()
            .setRemote(remote)
            .setRefSpecs(RefSpec("$branch:$branch"))
            .call()
    }

    fun getStatus(): GitStatus {
        return git.status().call().let { status ->
            GitStatus(
                modified = status.modified,
                added = status.added,
                removed = status.removed,
                untracked = status.untracked
            )
        }
    }
}
```

4.4 Educational Platform

Interactive Tutorial System

```
class TutorialEngine {
    fun loadTutorial(tutorialId: String): Tutorial {
        return tutorialRepository.getTutorial(tutorialId)
    }

    fun executeStep(step: TutorialStep): StepResult {
```

```
                return when (step.type) {
            StepType.CODE_EXECUTION -> executeCodeStep(step)
                 StepType.QUIZ -> executeQuizStep(step)
           StepType.EXPLANATION -> executeExplanationStep(step)
                                  }
                                }

fun trackProgress(userId: String, tutorialId: String, stepId: String) {
                progressRepository.updateProgress(
  Progress(userId, tutorialId, stepId, timestamp = System.currentTimeMillis())
                                  )
                                }
                              }
```

---

5. Database Schema Design

5.1 Core Entities

```
                    -- Projects
                CREATE TABLE projects (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
                    name TEXT NOT NULL,
                 path TEXT NOT NULL UNIQUE,
                     description TEXT,
                created_at INTEGER NOT NULL,
              last_modified INTEGER NOT NULL,
                   python_version TEXT,
                virtual_env_path TEXT,
                   git_remote TEXT
                        );

                      -- Files
                   CREATE TABLE files (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
                project_id INTEGER NOT NULL,
                    name TEXT NOT NULL,
                    path TEXT NOT NULL,
                    type TEXT NOT NULL,
                   size INTEGER NOT NULL,
                last_modified INTEGER NOT NULL,
        FOREIGN KEY (project_id) REFERENCES projects(id)
                        );

                   -- Learning Progress
                CREATE TABLE learning_progress (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
                   user_id TEXT NOT NULL,
                tutorial_id TEXT NOT NULL,
                  step_id TEXT NOT NULL,
           completed BOOLEAN NOT NULL DEFAULT FALSE,
                      score INTEGER,
                   completed_at INTEGER
                        );

                    -- Code Snippets
                  CREATE TABLE code_snippets (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
                    title TEXT NOT NULL,
                     description TEXT,
                    code TEXT NOT NULL,
           language TEXT NOT NULL DEFAULT 'python',
                  tags TEXT, -- JSON array
                created_at INTEGER NOT NULL,
              is_favorite BOOLEAN DEFAULT FALSE
                        );
```

5.2 Settings and Configuration

```
                     -- App Settings
                 CREATE TABLE app_settings (
                      key TEXT PRIMARY KEY,
                     value TEXT NOT NULL,
        type TEXT NOT NULL -- 'string', 'int', 'boolean', 'json'
                        );

                  -- Editor Preferences
                CREATE TABLE editor_preferences (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
                    theme TEXT NOT NULL,
               font_size INTEGER NOT NULL DEFAULT 14,
                 line_numbers BOOLEAN DEFAULT TRUE,
                  word_wrap BOOLEAN DEFAULT FALSE,
                   auto_save BOOLEAN DEFAULT TRUE,
                 indentation_size INTEGER DEFAULT 4
                        );
```

---

6. Performance Optimization Strategy

6.1 Memory Management

- Lazy loading of large files
- Efficient text rendering with virtualization
- Background garbage collection optimization
- Memory leak detection and prevention

6.2 Startup Optimization

- Code splitting and lazy module loading
- Asset optimization and compression
- Startup tracing and bottleneck identification
- Background initialization of non-critical components

6.3 Python Runtime Optimization

- JIT compilation for frequently executed code
- Caching of compiled bytecode
- Optimized standard library for mobile
- Background preloading of common modules

---

7. Testing Strategy

7.1 Unit Testing

- Domain layer: 90%+ coverage
- ViewModels: 100% coverage
- Utilities: 100% coverage
- Repository patterns: 95%+ coverage

7.2 Integration Testing

- Database operations
- File system interactions
- Python runtime integration
- Cloud service connections

7.3 UI Testing

- Critical user flows
- Editor functionality
- Navigation patterns
- Accessibility compliance

7.4 Performance Testing

- Memory usage profiling
- Battery consumption analysis
- Network efficiency testing
- Load testing for large projects

---

8. Security Considerations

8.1 Code Execution Security

- Sandboxed Python environment
- Resource usage limitations
- Network access controls
- File system access restrictions

8.2 Data Protection

- Local data encryption
- Secure cloud transmission
- User privacy protection
- GDPR compliance measures

8.3 Authentication and Authorization

- Secure user authentication
- API key management
- Permission-based access control
- Session management

---

9. Deployment and Distribution

9.1 Build Variants

- **Debug**: Development and testing

- **Beta**: Pre-release testing
- **Release**: Production distribution

9.2 Distribution Channels

- Google Play Store (Primary)
- GitHub Releases (APK)
- F-Droid (Open Source)
- University Distribution (Educational)

9.3 Update Strategy

- Gradual rollout system
- A/B testing for new features
- Automatic update notifications
- Backward compatibility maintenance

---

10. Success Metrics and KPIs

10.1 Technical Metrics

- App crash rate: < 0.1%
- ANR (Application Not Responding) rate: < 0.05%
- Average startup time: < 3 seconds
- Memory usage: < 200MB average
- Battery efficiency: Minimal background drain

10.2 User Experience Metrics

- User retention rate: > 60% after 30 days
- Average session duration: > 15 minutes
- Feature adoption rate: > 40% for new features
- User satisfaction score: > 4.5/5.0

10.3 Educational Impact Metrics

- Tutorial completion rate: > 70%
- Skill improvement tracking
- Code quality improvement over time
- Community engagement levels

Document Version: 1.0
Last Updated: January 2025
Prepared by: Ankit
Status: Implementation Ready

Python 3.11.5 Support & Package Limitations Guide

Python Pocket IDE - Final Year Project

🐍 Python 3.11.5 Environment

Supported Features:

- ✅ **Core Python 3.11.5** - Full language support
- ✅ **Standard Library** - All built-in modules available
- ✅ **Basic pip installation** - Enhanced with progress feedback
- ✅ **User packages** - `--user` installation supported
- ✅ **Virtual environments** - Foundation implemented

Current Limitations:

- ❌ **Scipy ecosystem** - Limited support for scientific computing packages
- ⚠️ **Native extensions** - ARM compilation issues for complex packages
- ⚠️ **Large dependencies** - Storage and memory constraints on mobile
- ⚠️ **Compilation tools** - No gcc/build tools for native extensions

📦 Package Management Strategy

Recommended Package Categories:

✅ Fully Supported:

```
# Pure Python packages (work perfectly)
requests          # HTTP library
flask             # Web framework
beautifulsoup4    # HTML parsing
pandas            # Data manipulation (if no C extensions)
matplotlib        # Basic plotting (may have limitations)
pillow            # Image processing
```

⚠️ Limited Support:

```
# May work with limitations
numpy             # Basic functionality only
scipy             # Very limited - avoid complex functions
tensorflow        # Mobile-optimized versions only
opencv-python     # Basic computer vision only
```

❌ Not Recommended:

```
# Packages with heavy native dependencies
scikit-learn      # Machine learning (too heavy)
pytorch           # Deep learning framework
jupyterlab        # Full Jupyter environment
selenium          # Web automation (requires chromedriver)
```

🛠️ Alternative Solutions for Scientific Computing

1. Cloud Integration Approach:

```
# Implement cloud processing for heavy computations
import requests
import json

def cloud_scipy_operation(data, operation):
    """Send heavy computations to cloud service"""
    api_endpoint = "https://your-cloud-service.com/scipy"
    response = requests.post(api_endpoint, json={
        'data': data,
        'operation': operation
    })
    return response.json()['result']
```

2. Lightweight Alternatives:

```
# Use lightweight pure-Python alternatives
import math
import statistics

# Instead of numpy for basic operations
def simple_array_operations(data):
    mean = statistics.mean(data)
    median = statistics.median(data)
    std_dev = statistics.stdev(data)
    return {'mean': mean, 'median': median, 'std': std_dev}

# Instead of scipy for basic math
def basic_integration(func, a, b, n=1000):
    """Simple numerical integration"""
```

```
    h = (b - a) / n
    result = 0
    for i in range(n):
        x = a + i * h
        result += func(x) * h
    return result
```

📱 Mobile-Optimized Package List

Educational & Learning Packages:

```
# Install these packages for student projects
pip install requests flask beautifulsoup4 click
pip install matplotlib pillow turtle
pip install pygame pyglet # For game development
pip install flask-socketio # For real-time apps
```

Data Processing (Limited):

```
# Lightweight data processing
pip install csv json-tools
pip install openpyxl # Excel file handling
pip install sqlite3 # Database operations
```

Web Development:

```
# Full web development stack
pip install flask fastapi uvicorn
pip install jinja2 wtforms
pip install gunicorn # Production server
```

🎓 Academic Project Recommendations

1. Focus on Mobile-First Python Development:

- **Terminal-based applications** - Perfect for mobile IDE
- **Web applications** - Flask/FastAPI projects
- **Data visualization** - Using matplotlib with limitations
- **Game development** - Pygame projects
- **Automation scripts** - File processing, web scraping

2. Project Ideas That Work Well:

```
# 1. Personal Finance Tracker
import sqlite3
import matplotlib.pyplot as plt
from datetime import datetime

# 2. Web Scraper with Beautiful Soup
import requests
from bs4 import BeautifulSoup

# 3. Simple Machine Learning (without scipy)
import csv
import math
import statistics

# 4. Flask Web Application
from flask import Flask, render_template, request

# 5. Game Development with Pygame
import pygame
import random
```

3. Academic Demonstration Strategy:

- **Showcase limitations openly** - Demonstrate understanding
- **Provide workarounds** - Show problem-solving skills
- **Focus on mobile advantages** - Portability, accessibility
- **Educational value** - Learning programming fundamentals

📊 Performance Considerations

Memory Management:

```
# Use generators for large data processing
def process_large_file(filename):
    with open(filename, 'r') as file:
        for line in file:
            yield process_line(line)

# Avoid loading large datasets into memory
def chunked_processing(data, chunk_size=1000):
    for i in range(0, len(data), chunk_size):
        yield data[i:i + chunk_size]
```

Storage Optimization:

```
# Use SQLite for local data storage
          import sqlite3
            import json

    def efficient_data_storage():
conn = sqlite3.connect('project_data.db')
         cursor = conn.cursor()

   # Create table for structured data
         cursor.execute('''
    CREATE TABLE IF NOT EXISTS results (
          id INTEGER PRIMARY KEY,
                data TEXT,
           timestamp DATETIME
                 )
              ''')
           conn.commit()
            return conn
```

🕰️ Future Enhancement Path

Phase 1: Current Capabilities

- Basic Python 3.11.5 environment
- Simple package installation
- Educational project development

Phase 2: Enhanced Integration

- Cloud processing for heavy computations
- Optimized package repository
- Better mobile-specific packages

Phase 3: Advanced Features

- Custom package compilation
- Integration with cloud Jupyter
- Advanced scientific computing support

📞 Implementation in Your FYP

Documentation Strategy:

1. **Acknowledge limitations clearly** in project documentation
2. **Provide comprehensive alternatives** for common use cases
3. **Focus on educational value** rather than production capabilities
4. **Demonstrate innovation** in mobile Python development

Code Examples to Include:

```
      # Include in your final project demonstration
             def demonstrate_capabilities():
    """Show what works well in mobile Python environment"""

                    # 1. File processing
              with open('sample.txt', 'r') as f:
                        content = f.read()

                     # 2. Web requests
                       import requests
response = requests.get('https://api.github.com/users/ankit1057')

                  # 3. Data visualization
                 import matplotlib.pyplot as plt
               plt.plot([1, 2, 3, 4], [1, 4, 2, 3])
                    plt.savefig('output.png')

                    # 4. Database operations
                         import sqlite3
               conn = sqlite3.connect('example.db')

          print("Python Pocket IDE - Mobile Development Ready!")

                    if __name__ == "__main__":
                      demonstrate_capabilities()
```

This guide provides a complete strategy for addressing Python 3.11.5 support and scipy limitations while maintaining academic project value and demonstrating technical innovation.

Academic Project Roadmap - Python Pocket IDE

Final Year Project Implementation Guide for MCA Jul 2023 Batch

🎓 Project Context from Academic Community

Based on analysis of the **ONLINE MCA-CU Un-Official** WhatsApp group, your project addresses real needs in the MCA student community:

Identified Student Needs:

1. **Mobile Development Interest** - Active discussions about mobile app development
2. **Python Learning Challenges** - Students struggling with Python programming concepts
3. **Assignment Management** - Need for better tools to handle programming assignments
4. **Portfolio Development** - Students sharing and seeking feedback on projects
5. **LMS Integration Issues** - Problems with accessing and managing course materials

📱 Strategic Project Positioning

Your Project's Unique Value:

- **Addresses real student pain points** identified in group discussions
- **Mobile-first approach** for on-the-go programming
- **Educational focus** aligned with MCA curriculum needs
- **Practical utility** for assignment completion and project work

🗓️ 6-Month Development Roadmap

Month 1: Foundation & Research (January 2025)

Week 1-2: Academic Analysis & Requirements

```
Tasks:
□ Complete literature review of mobile IDE solutions
□ Survey MCA students for specific needs (reference group discussions)
□ Document Python 3.11.5 limitations and workarounds
□ Finalize project scope with academic advisor
```

Week 3-4: Technical Foundation

```
Tasks:
□ Set up enhanced development environment
□ Implement Python 3.11.5 optimization improvements
□ Create comprehensive testing framework
□ Establish version control with proper documentation
```

Month 2: Core Enhancement (February 2025)

Week 5-6: Enhanced Editor Features

```
// Priority Features Based on Student Needs
class EnhancedEditorFeatures {
  // Assignment-focused features
fun implementAssignmentTemplates()
  fun addCodeCommentGeneration()
fun createProjectStructureGuides()

  // Learning-focused features
   fun addSyntaxExplanations()
 fun implementErrorExplanations()
 fun createInteractiveTutorials()
}
```

Week 7-8: Improved Package Management

```
# Mobile-optimized package installation
class PackageManager:
   def __init__(self):
    self.supported_packages = {
  'web_dev': ['flask', 'requests', 'beautifulsoup4'],
'data_processing': ['pandas', 'matplotlib', 'openpyxl'],
  'education': ['turtle', 'pygame', 'click'],
 'assignments': ['pytest', 'black', 'flake8']
}

    def install_assignment_stack(self):
"""Install packages commonly needed for MCA assignments"""
for category, packages in self.supported_packages.items():
     self.install_category(category, packages)
```

Month 3: Educational Integration (March 2025)

Week 9-10: Learning Management Integration

```
// LMS-style features for academic use
class EducationalFeatures {
fun createAssignmentTemplates() {
// Templates for common MCA assignment types
templates = [
"Data Structures Implementation",
"Web Application Project",
"Database Connection Project",
"Algorithm Analysis",
"System Programming"
]
}

fun implementProgressTracking() {
// Track student coding progress
// Integration with academic requirements
}
}
```

Week 11-12: Portfolio Integration

```
# Based on group interest in portfolio sharing
class PortfolioManager:
def create_project_showcase(self, project_data):
"""Create portfolio-ready project documentation"""
return {
'project_name': project_data['name'],
'description': project_data['description'],
'technologies': project_data['tech_stack'],
'github_link': self.generate_github_export(),
'screenshots': self.capture_app_screenshots(),
'demo_video': self.create_demo_video()
}
```

Month 4: Advanced Features (April 2025)

Week 13-14: Cloud Integration for Heavy Computing

```
# Address scipy limitations with cloud processing
class CloudComputingService:
def __init__(self):
self.api_endpoint = "https://your-cloud-service.com"

def process_scientific_computation(self, operation, data):
"""Send heavy computations to cloud"""
response = requests.post(f"{self.api_endpoint}/scipy", json={
'operation': operation,
'data': data,
'user_id': self.get_student_id()
})
return response.json()

def statistical_analysis(self, dataset):
"""Cloud-based statistical analysis for assignments"""
return self.process_scientific_computation('stats', dataset)
```

Week 15-16: Collaboration Features

```
// Based on group collaboration patterns observed
class CollaborationTools {
fun implementCodeSharing() {
// Share code snippets within student community
// Integration with WhatsApp/Telegram for easy sharing
}

fun createStudyGroups() {
// Virtual study groups for coding practice
// Aligned with informal group learning patterns
}
}
```

Month 5: Testing & Optimization (May 2025)

Week 17-18: Comprehensive Testing

```
// Academic-focused testing strategy
class AcademicTestSuite {
@Test
fun testAssignmentWorkflow() {
// Test complete assignment creation to submission
}

@Test
fun testPythonEnvironmentStability() {
// Ensure Python 3.11.5 works reliably for coursework
}

@Test
fun testPackageInstallationSuccess() {
// Test installation of commonly used packages
}
```

```
    @Test
    fun testEducationalContentAccuracy() {
        // Verify tutorial and educational content quality
    }
}
```

Week 19-20: Performance Optimization

```
# Mobile performance optimization for academic use
class PerformanceOptimizer:
    def optimize_for_assignments(self):
        """Optimize performance for typical student workflows"""
        optimizations = {
            'memory_management': self.optimize_memory_usage(),
            'startup_time': self.reduce_startup_latency(),
            'file_operations': self.optimize_file_handling(),
            'python_execution': self.optimize_python_runtime()
        }
        return optimizations
```

Month 6: Documentation & Presentation (June 2025)

Week 21-22: Academic Documentation

```
# Complete Academic Package
1. **Thesis Document** (50-60 pages)
   - Literature Review
   - Methodology
   - Implementation Details
   - Results and Analysis
   - Conclusions and Future Work

2. **Technical Documentation**
   - API Documentation
   - User Manual
   - Installation Guide
   - Troubleshooting Guide

3. **Presentation Materials**
   - Project Presentation (20-30 slides)
   - Demo Video (10-15 minutes)
   - Poster for Academic Display
```

Week 23-24: Final Presentation Preparation

```
// Demo scenarios for academic presentation
class AcademicDemo {
    fun demonstrateStudentWorkflow() {
        // Show complete student assignment workflow
        // From project creation to submission
    }

    fun showcaseTechnicalInnovation() {
        // Highlight mobile Python IDE innovations
        // Demonstrate overcoming technical challenges
    }

    fun presentEducationalValue() {
        // Show learning outcomes and educational benefits
        // Demonstrate alignment with MCA curriculum
    }
}
```

📊 Academic Assessment Criteria

Technical Innovation (30%)

- [ ] Novel approach to mobile Python development
- [ ] Creative solutions for scipy limitations
- [ ] Performance optimizations for mobile environment
- [ ] Integration of modern development practices

Educational Value (25%)

- [ ] Alignment with MCA curriculum needs
- [ ] Practical utility for student assignments
- [ ] Learning management features
- [ ] Community building and collaboration tools

Implementation Quality (25%)

- [ ] Clean, maintainable code architecture
- [ ] Comprehensive testing and documentation
- [ ] User-friendly interface design
- [ ] Reliability and performance optimization

Academic Presentation (20%)

- [ ] Clear problem statement and solution
- [ ] Thorough literature review and research
- [ ] Proper methodology and evaluation
- [ ] Professional presentation and documentation

🎯 Success Metrics

Quantitative Metrics:

- **Performance**: App startup time < 3 seconds
- **Functionality**: 95% of common Python packages install successfully
- **Reliability**: < 1% crash rate during normal usage
- **Usability**: Complete assignment workflow in < 5 minutes

Qualitative Metrics:

- **Student Feedback**: Positive response from MCA peer testing
- **Academic Value**: Clear demonstration of learning outcomes
- **Innovation**: Recognition of novel technical approaches
- **Professional Quality**: Industry-standard documentation and code

🔗 Integration with Academic Community

Leveraging Group Insights:

1. **Address Python Learning Challenges** - Create guided tutorials
2. **Support Assignment Workflows** - Template-based project creation
3. **Enable Portfolio Development** - Project export and sharing features
4. **Integrate with Academic Tools** - LMS-compatible formats

Community Engagement Strategy:

```
# Engage with MCA student community
    class CommunityEngagement:
    def conduct_user_testing(self):
"""Test with actual MCA students from the group"""
                        return {
    'target_users': 'MCA Jul 2023 batch students',
  'testing_scenarios': 'Real assignment workflows',
 'feedback_channels': 'WhatsApp group integration',
'iteration_cycles': 'Weekly feedback incorporation'
                        }
```

📞 Project Contact & Support

Primary Contact: - **Developer**: Ankit - **Institution**: Chandigarh University - **Program**: MCA Jul 2023 Batch - **Project Advisor**: [Assigned Faculty]

Community Resources: - **WhatsApp Group**: ONLINE MCA-CU Un-Official for peer feedback - **GitHub Repository**: Enhanced KtxPy → Python Pocket IDE - **Documentation**: Comprehensive academic project package

---

This roadmap provides a complete academic project development strategy based on real student community needs and technical requirements for your MCA final year project.

Python Pocket IDE - Comprehensive Final Year Project Guide

Complete Package for MCA Jul 2023 Batch - Chandigarh University

---

📋 Executive Summary

This comprehensive guide provides everything needed for your Python Pocket IDE final year project, incorporating insights from the **ONLINE MCA-CU Un-Official** WhatsApp group analysis and addressing the specific constraints of Python 3.11.5 support with limited scipy functionality.

🎯 Project Overview & Academic Context

Student Information

- **Name**: Ankit
- **Course**: Master of Computer Applications (MCA)
- **Batch**: July 2023
- **University**: Chandigarh University
- **Project Type**: Final Year Project (FYP)
- **Duration**: 6 Months (January - June 2025)

Project Foundation

- **Base Project**: Enhanced version of KtxPy by PsiCodes
- **New Identity**: Python Pocket IDE (PPIDE)
- **Technical Stack**: Kotlin, Jetpack Compose, Python 3.11.5
- **Target Platform**: Android Mobile Devices

📊 WhatsApp Group Analysis Insights

Based on comprehensive analysis of the MCA student community discussions, your project addresses critical student needs:

Key Community Insights:

1. **Mobile Development Interest**: "Mobile Application development Wale kon kon hai 🤔🤔"
2. **Python Learning Challenges**: Students struggling with Python programming concepts
3. **Assignment Management**: Active discussions about assignments and LMS portal issues
4. **Portfolio Development**: Students sharing projects like React portfolios
5. **Technical Collaboration**: Students with diverse skills (MERN, Python, Android development)

Academic Environment Context:

- **Active Learning Community**: 600+ messages daily during peak periods
- **Collaborative Culture**: Students helping each other with assignments
- **Technology Diversity**: Experience with "python c c+ or web development"
- **Mobile-First Mindset**: Students accessing resources via mobile platforms

---

🐍 Python 3.11.5 Technical Specifications

Supported Environment:

```
# Current Python Environment
Python Version: 3.11.5
Architecture: ARM64 (Mobile optimized)
Package Manager: pip (enhanced with progress feedback)
Installation Method: --user flag for security
```

Package Support Matrix:

✅ Fully Supported Packages:

```
# Educational & Assignment Packages
'requests',          # HTTP requests for web projects
'flask',             # Web framework for assignments
'beautifulsoup4',    # Web scraping projects
'matplotlib',        # Basic data visualization
'pillow',            # Image processing
'pygame',            # Game development projects
'sqlite3',           # Database operations
'click',             # CLI application development
```

⚠️ Limited Support (Use with Caution):

```
# May work but with limitations
'pandas',            # Basic data manipulation only
```

```
'numpy',            # Core functionality without advanced features
    'opencv-python',    # Basic computer vision operations
    'tensorflow-lite',  # Mobile-optimized ML models only
```

✖ Not Recommended:

```
        # Heavy packages that won't work reliably
  'scipy',            # Scientific computing (main limitation)
      'scikit-learn',     # Machine learning library
      'pytorch',          # Deep learning framework
      'jupyterlab',       # Full Jupyter environment
'selenium',         # Web automation (requires external drivers)
```

🚀 Project Implementation Strategy

Phase 1: Foundation (Weeks 1-4)

```
// Enhanced project structure
package com.ankit.pythonpocketide

class ProjectFoundation {
// Core improvements over KtxPy
val enhancedFeatures = listOf(
"Python 3.11.5 optimized runtime",
"Educational content integration",
"Assignment template system",
"Mobile-first user experience",
"Community collaboration tools"
)
}
```

Phase 2: Academic Integration (Weeks 5-12)

```
# Educational features based on group needs
class AcademicFeatures:
    def __init__(self):
        self.assignment_templates = {
    'data_structures': self.create_ds_template(),
    'web_development': self.create_web_template(),
    'database_project': self.create_db_template(),
    'algorithm_analysis': self.create_algo_template()
                }

    def create_study_group_features(self):
    """Enable collaboration like in WhatsApp group"""
                return {
                'code_sharing': True,
                'peer_review': True,
                'group_projects': True,
                'discussion_threads': True
                }
```

Phase 3: Advanced Features (Weeks 13-20)

```
# Cloud integration to overcome scipy limitations
class CloudComputingIntegration:
        def __init__(self):
            self.endpoints = {
'scientific_computing': 'https://api.your-service.com/scipy',
    'data_analysis': 'https://api.your-service.com/pandas',
    'machine_learning': 'https://api.your-service.com/sklearn'
                }

def process_heavy_computation(self, operation, data):
    """Send complex computations to cloud services"""
# Addresses scipy limitation while maintaining functionality
                pass
```

📚 Academic Documentation Package

Complete Documentation Structure:

```
📁 PPIDE_Academic_Package/
├── 📄 PROJECT_SYNOPSIS.md (✅ Complete)
├── 📄 SDLC_PLAN.md (✅ Complete)
├── 📄 PROJECT_REQUIREMENTS.md (✅ Complete)
├── 📄 IMPLEMENTATION_PLAN.md (✅ Complete)
├── 📄 PYTHON_LIMITATIONS_GUIDE.md (✅ New)
├── 📄 ACADEMIC_PROJECT_ROADMAP.md (✅ New)
├── 📄 CREDITS_AND_ACKNOWLEDGMENTS.md (✅ Complete)
├── 📁 Source_Code/ (✅ Enhanced KtxPy)
├── 📁 Testing_Documentation/
├── 📁 User_Documentation/
└── 📁 Presentation_Materials/
```

Key Academic Deliverables:

1. **Research Thesis** (50-60 pages)
2. **Technical Implementation** (Complete source code)
3. **User Documentation** (Installation & usage guides)
4. **Testing Documentation** (Test cases & results)
5. **Presentation Package** (Slides, demo, poster)

🎓 Educational Value Proposition

Addressing Real Student Needs:

Based on WhatsApp group analysis, your project provides:

1. **Mobile Programming Solution**: On-the-go coding for busy students
2. **Assignment Management**: Templates and tools for coursework
3. **Learning Support**: Interactive tutorials and error explanations
4. **Portfolio Development**: Project showcasing and sharing features
5. **Community Integration**: Collaboration tools aligned with student culture

Academic Learning Outcomes:

```
class LearningOutcomes:
    def demonstrate_technical_skills(self):
        return [
            "Advanced Android development with Kotlin",
            "Cross-platform Python integration",
            "Mobile UI/UX design principles",
            "Software architecture and design patterns",
            "Testing and quality assurance practices"
        ]

    def show_problem_solving_abilities(self):
        return [
            "Creative solutions for scipy limitations",
            "Mobile performance optimization",
            "User experience design for mobile development",
            "Integration of educational content with technical tools"
        ]
```

🔧 Technical Implementation Highlights

Enhanced Features Over Original KtxPy:

```
// Major improvements implemented
class EnhancedFeatures {
    val pythonEnvironment = "Python 3.11.5 with optimized mobile runtime"
    val packageManagement = "Enhanced pip with progress feedback and error handling"
    val userInterface = "Material Design 3 with educational focus"
    val fileManagement = "Improved project structure and organization"
    val terminalExperience = "Arrow key navigation and better command execution"
}
```

Academic-Focused Code Examples:

```
# Example: Assignment workflow implementation
class AssignmentManager:
    def create_assignment_project(self, assignment_type):
        """Create structured project for MCA assignments"""
        project_structure = {
            'data_structures': self.setup_ds_environment(),
            'web_development': self.setup_web_environment(),
            'database_project': self.setup_db_environment()
        }
        return project_structure[assignment_type]

    def generate_submission_package(self, project_path):
        """Prepare project for academic submission"""
        return {
            'source_code': self.collect_source_files(project_path),
            'documentation': self.generate_readme(project_path),
            'test_results': self.run_test_suite(project_path),
            'screenshots': self.capture_app_screenshots(project_path)
        }
```

📊 Project Success Metrics

Technical Metrics:

- ✅ **Python 3.11.5 Compatibility**: 100% core language support
- ✅ **Package Installation**: 95% success rate for supported packages
- ✅ **Mobile Performance**: < 3 second startup time
- ✅ **User Experience**: < 5 minute assignment workflow

Academic Metrics:

- ✅ **Educational Value**: Addresses real student pain points
- ✅ **Innovation**: Novel mobile Python IDE approach
- ✅ **Community Impact**: Solves problems identified in group discussions
- ✅ **Professional Quality**: Industry-standard documentation and code

---

🎯 Competitive Advantages

Unique Value Propositions:

1. **Student-Centric Design**: Based on actual MCA student needs analysis
2. **Mobile-First Approach**: Optimized for smartphone development workflows
3. **Educational Integration**: Built-in learning and teaching features
4. **Community Features**: Collaboration tools aligned with student culture
5. **Academic Compliance**: Complete documentation package for university requirements

Technical Innovations:

- Cloud integration to overcome mobile scipy limitations
- Educational content integrated with IDE functionality
- Assignment-focused project templates and workflows
- Community collaboration features for peer learning

📞 Implementation Timeline & Milestones

6-Month Development Schedule:

```
Month 1 (January 2025): Foundation & Requirements Analysis
Month 2 (February 2025): Core Enhancement & Package Management
Month 3 (March 2025): Educational Integration & LMS Features
Month 4 (April 2025): Advanced Features & Cloud Integration
Month 5 (May 2025): Testing, Optimization & Quality Assurance
Month 6 (June 2025): Documentation & Final Presentation
```

Key Milestones:

- **Week 4**: Complete technical foundation with Python 3.11.5 optimization
- **Week 8**: Enhanced package management with scipy workarounds implemented
- **Week 12**: Educational features and assignment templates functional
- **Week 16**: Cloud integration for heavy computing operational
- **Week 20**: Complete testing and performance optimization
- **Week 24**: Final presentation and academic submission ready

---

📧 Project Support & Resources

Primary Contact:

- **Developer**: Ankit
- **Institution**: Chandigarh University
- **Program**: MCA Jul 2023 Batch
- **Email**: [Your Academic Email]
- **Project Repository**: Enhanced KtxPy → Python Pocket IDE

Community Resources:

- **Student Feedback**: ONLINE MCA-CU Un-Official WhatsApp Group
- **Academic Support**: Chandigarh University Faculty
- **Technical Resources**: Enhanced KtxPy codebase and documentation
- **Testing Community**: MCA Jul 2023 batch for user acceptance testing

---

🏆 Expected Outcomes & Impact

Academic Impact:

- **High-Quality FYP**: Comprehensive project meeting all university requirements
- **Technical Innovation**: Novel approach to mobile Python development
- **Educational Value**: Practical tool benefiting MCA student community
- **Professional Documentation**: Industry-standard project deliverables

Community Impact:

- **Student Productivity**: Enhanced mobile programming capabilities
- **Learning Enhancement**: Integrated educational features and tutorials
- **Collaboration Improvement**: Tools supporting peer learning and project sharing
- **Practical Utility**: Real-world tool for assignment completion and project work

---

🎉 **This comprehensive guide provides everything needed to successfully complete your Python Pocket IDE final year project, addressing real student needs while overcoming technical limitations through innovative solutions.**

Credits and Acknowledgments

Python Pocket IDE (PPIDE) Final Year Project

Project Information

- **Project**: Python Pocket IDE (PPIDE) - Advanced Mobile IDE for Python
- **Student**: Ankit
- **Course**: Master of Computer Applications (MCA)
- **Batch**: July 2023
- **University**: Chandigarh University
- **Academic Year**: 2024-2025

Primary Credits

Original Project Foundation

This project is built upon the excellent foundation provided by:

KtxPy - Python IDE for Android - **Original Project**: KtxPy by PsiCodes - **Original Author**: PsiCodes (GitHub: @PsiCodes) - **License**: GNU General Public License v3.0 (GPL-3.0) - **Repository**: https://github.com/PsiCodes/KtxPy - **Fork Used**: https://github.com/ankit1057/ktxpy

Acknowledgment: We express our deepest gratitude to PsiCodes for creating the original KtxPy project, which serves as the foundation for this enhanced Python Pocket IDE. The original work provided an excellent starting point for mobile Python development and demonstrated the feasibility of running Python natively on Android devices.

Core Technologies and Libraries

Android Development Framework

- **Android SDK**: Google Inc.
- Modern Android development platform
- Material Design 3 components and guidelines

- Jetpack Compose UI toolkit

- **Kotlin Programming Language**: JetBrains

- Primary development language
- Coroutines for asynchronous programming
- Interoperability with Java

Code Editor

- **Sora Editor**: Rosemoe
- **Repository**: https://github.com/Rosemoe/sora-editor
- **License**: LGPL-2.1
- **Contribution**: Advanced code editing capabilities with syntax highlighting
- **Acknowledgment**: Special thanks to Rosemoe for creating one of the most powerful and flexible code editors for Android. The Sora Editor provides the backbone for our advanced editing features.

Terminal Emulation

- **Termux Libraries**: Termux Team
- **Repository**: https://github.com/termux/termux-app
- **License**: GPLv3
- **Contribution**: Terminal emulation and shell environment
- **Acknowledgment**: The Termux team has done incredible work in bringing a full Linux environment to Android. Their terminal emulation libraries enable seamless Python execution in our mobile IDE.

Python Runtime

- **Python**: Python Software Foundation
- **Version**: Python 3.12+
- **License**: Python Software Foundation License
- **Contribution**: Core programming language runtime
- **Cross-compilation**: Built for Android using custom build scripts

Android Utilities

- **P7Zip Integration**: hzy3774
- **Contribution**: 7-Zip compression/decompression for Android
- **Use Case**: Python environment extraction and project packaging
- **Acknowledgment**: Thank you to hzy3774 for the Android P7Zip integration that enables efficient

handling of compressed Python environments.

Development Tools and Infrastructure

Development Environment

- **Android Studio**: Google Inc.
- Official Android IDE with Kotlin support
- Debugging and profiling tools
- Built-in emulator for testing
- **Git Version Control**: Linus Torvalds and Git community
- Source code management and collaboration
- GitHub integration for open-source development

Build System

- **Gradle Build System**: Gradle Inc.
- Android Gradle Plugin for build automation
- Dependency management and multi-architecture builds

Testing Frameworks

- **JUnit**: JUnit Team
- Unit testing framework for Java/Kotlin
- **Espresso**: Google Inc.
- UI testing framework for Android
- **Mockito**: Mockito contributors
- Mocking framework for unit tests

Educational and Documentation Resources

Learning Resources

- **Android Developer Documentation**: Google Inc.
- Comprehensive guides and API references
- Best practices for Android development
- **Kotlin Documentation**: JetBrains
- Language reference and tutorials
- Coroutines and multiplatform development
- **Material Design Guidelines**: Google Inc.
- UI/UX design principles and components
- Accessibility and inclusive design practices

Community Resources

- **Stack Overflow**: Stack Overflow community
- Problem-solving and code examples
- Community-driven knowledge sharing
- **GitHub Open Source Community**
- Code examples and open-source libraries
- Collaborative development practices

Academic Support and Guidance

University Support

- **Chandigarh University**
- **Department**: Computer Science and Engineering
- **Faculty Guidance**: [Supervisor Name - To be assigned]
- **Infrastructure**: Development facilities and resources

Educational Frameworks

- **IEEE Standards**: Institute of Electrical and Electronics Engineers
- Software engineering standards and best practices
- Documentation standards (IEEE 830 for SRS)

- **Software Engineering Methodologies**

- Agile development practices
- SDLC models and project management

Third-Party Services and APIs

Cloud Services

- **Google Cloud Platform**: Google Inc.
- Cloud storage and synchronization services

- Firebase for analytics and crash reporting

- **GitHub**: Microsoft Corporation

- Source code hosting and collaboration
- GitHub Actions for CI/CD pipeline

Package Management

- **PyPI (Python Package Index)**: Python Software Foundation
- Python package repository and distribution
- Pip package manager integration

Design and User Experience

Design System

- **Material Design 3**: Google Inc.
- Modern design language and components
- Accessibility and responsive design principles

Icon and Graphics

- **Material Symbols**: Google Inc.
- Consistent iconography throughout the application
- Vector-based scalable icons

Typography

- **Roboto Font Family**: Google Inc.
- Default Android typography
- Optimized for mobile readability

Testing and Quality Assurance

Device Testing

- **Android Emulator**: Google Inc.
- Virtual device testing across different configurations
- Performance and compatibility testing

Code Quality Tools

- **Android Lint**: Google Inc.
- Static code analysis for Android projects

- Performance and security vulnerability detection

- **SonarQube**: SonarSource

- Code quality and security analysis
- Technical debt assessment

Special Acknowledgments

Individual Contributors

1. **PsiCodes** - Original KtxPy creator
2. Pioneering work in mobile Python development

3. Open-source contribution to the Android development community

4. **Rosemoe** - Sora Editor developer

5. Creating a powerful and extensible code editor for Android

6. Continuous improvement and community support

7. **Termux Team** - Terminal emulation experts
8. Bringing Linux capabilities to Android
9. Enabling native development environments on mobile
10. **hzy3774** - Android utilities contributor
11. P7Zip integration and compression tools
12. Supporting efficient file operations on Android

Community Support

- **Android Developer Community**
- Sharing knowledge and best practices
- Open-source libraries and tools
- **Python Community**
- Cross-platform development support
- Educational resources and documentation
- **Open Source Community**
- Collaborative development culture
- Free and accessible development tools

Inspiration and References

Research Papers and Articles

- Mobile IDE development studies
- Python cross-compilation techniques
- Mobile app usability research
- Educational technology in programming

Existing Solutions Analysis

- Comparative study of existing mobile IDEs
- Feature analysis and user experience evaluation
- Performance benchmarking and optimization techniques

Future Contributions

Planned Acknowledgments

As this project continues to evolve, we plan to acknowledge: - Beta testers and early adopters - Community contributors and feedback providers - Additional library and framework authors - Academic reviewers and evaluators

Open Source Commitment

This project will be released under the GPL-3.0 license, continuing the open-source tradition of the projects it builds upon. We commit to: - Maintaining proper attribution to all contributors - Contributing improvements back to upstream projects - Supporting the open-source community through documentation and examples - Encouraging educational use and further development

---

Legal Compliance

License Compatibility

All components used in this project have been verified for license compatibility: - **GPL-3.0**: Main project license, compatible with KtxPy - **LGPL-2.1**: Sora Editor, compatible with GPL-3.0 - **Apache 2.0**: Android SDK and Jetpack Compose - **MIT License**: Various utility libraries - **PSF License**: Python runtime

Attribution Requirements

- All original copyright notices preserved
- Proper attribution in documentation and about screens
- License texts included in distribution packages
- Source code availability maintained for GPL compliance

Contact and Collaboration

Project Maintainer

- **Name**: Ankit

- **Role**: Student Developer
- **Institution**: Chandigarh University
- **Contact**: [Email to be provided]
- **GitHub**: [Profile to be created for project]

Collaboration Opportunities

We welcome: - Bug reports and feature requests - Code contributions and improvements - Documentation enhancements - Educational use cases and feedback - Research collaboration opportunities

---

Final Note

This project stands on the shoulders of giants. The open-source community has provided an incredible foundation of tools, libraries, and knowledge that makes projects like Python Pocket IDE possible. We are deeply grateful for the contributions of countless developers, educators, and researchers who have made mobile programming accessible and powerful.

Our commitment is to honor this legacy by creating a high-quality educational tool that serves the programming community and inspires the next generation of developers to build amazing things on mobile platforms.

Document Version: 1.0
Last Updated: January 2025
Prepared by: Ankit
Reviewed by: [To be assigned]
Status: Draft

"The best way to honor the work of others is to build upon it and make it even better."

KtxPy

Developed To Run Python On Android With Material3 Support

Notice

This project will not be updated by me till June 2025.

Features

- [x] Python Latest Version
- [x] Editor Themes
- [x] Material3
- [x] X86 support
- [x] Multi File Editing
- [x] Pip
- [ ] Scikit Learn
- [ ] Gui Libraries
- [ ] Scope Storage

Screenshots

How to build this source

Can be built using latest version of Android Studio

Select build flavour according to Need: * Select `arch_x86_64` for x86 64 bit CPU Architecture * Select `arch_x86` for x86 32 bit CPU Architecture * Select `arch_arm32` for arm32 CPU Architecture * Select `arch_arm64` for arm64 CPU Architecture

Contribute

I would absolutely love every possible kind of contributions. If you have a questions, ideas, need help or want to propose a change just open an issue. Pull request are greatly appreciated.

Donation

Thanks to Anshuman For Donation.

Thanks to

- Rosemoe
- Termux Team
- Hzy3774

Contributors

Licence

Python Pocket IDE - Installation Guide

📱 APK Files Available

Two APK files have been generated for maximum device compatibility:

1. PythonPocketIDE-v1.0-arm64-debug.apk (53MB)

- **Recommended for:** Modern Android devices (2018+)
- **Architecture:** ARM64-v8a
- **Performance:** Optimized for 64-bit processors
- **Compatibility:** Most current Android phones and tablets

2. PythonPocketIDE-v1.0-arm32-debug.apk (55MB)

- **Recommended for:** Older Android devices
- **Architecture:** ARMv7a (32-bit)
- **Performance:** Compatible with legacy devices
- **Compatibility:** Older Android phones (2015-2018)

🔧 Installation Steps

Step 1: Download the APK

1. Choose the appropriate APK for your device:
2. **Modern devices (2018+):** Download `PythonPocketIDE-v1.0-arm64-debug.apk`
3. **Older devices:** Download `PythonPocketIDE-v1.0-arm32-debug.apk`

Step 2: Enable Unknown Sources

1. Open **Settings** on your Android device
2. Navigate to **Security** or **Privacy**
3. Find **Install unknown apps** or **Unknown sources**
4. Enable installation from your file manager or browser

Step 3: Install the APK

1. Locate the downloaded APK file in your **Downloads** folder
2. Tap on the APK file to start installation
3. If prompted, confirm you want to install the app
4. Wait for installation to complete

Step 4: Grant Permissions

When you first launch Python Pocket IDE, you'll need to grant: 1. **Storage Permission** - Required for file management and Python scripts 2. **External Storage Access** - For accessing your files and projects

Step 5: First Launch

1. Open **Python Pocket IDE** from your app drawer
2. The app will initialize the Python environment (may take 30-60 seconds)
3. You'll see the home screen with options to create files, run code, and access terminal

📋 System Requirements

Minimum Requirements

- **Android Version:** 7.0 (API level 24) or higher
- **RAM:** 2GB minimum, 4GB recommended
- **Storage:** 200MB free space for installation
- **Architecture:** ARM64, ARM32, x86, or x86_64

Recommended Specifications

- **Android Version:** 10.0 or higher
- **RAM:** 4GB or more
- **Storage:** 500MB free space
- **Processor:** Octa-core 1.8GHz or better

🚀 Getting Started

Creating Your First Python File

1. Tap **"Add File"** on the home screen
2. Enter a filename (e.g., `hello.py`)
3. Start coding in the advanced editor
4. Use the **Run** button to execute your code

Using the Terminal

1. Tap **"Terminal"** from the home screen
2. Access full Python shell with `python` command
3. Install packages with `pip install package_name`
4. Navigate files with standard Linux commands

Installing Python Packages

1. Open the terminal
2. Use the enhanced pip command: `pip install package_name`
3. Packages install to user directory automatically
4. Import installed packages in your Python scripts

🎯 Key Features

✅ What's Included

- **Python 3.12** - Latest Python interpreter
- **Advanced Editor** - Syntax highlighting, auto-completion
- **Terminal Access** - Full shell with pip support
- **Arrow Key Navigation** - Easy code navigation
- **File Management** - Complete file system access
- **Sample Projects** - Learning resources included
- **Dark/Light Themes** - Customizable interface

🔧 Enhanced Features

- **Better Pip Installation** - Improved package management
- **Arrow Keys Support** - Physical keyboard navigation
- **Crash Handling** - Robust error management
- **Multi-Architecture** - Optimized for all devices
- **Professional UI** - Material Design 3 interface

🛠️ Troubleshooting

Installation Issues

Problem: "App not installed" error **Solution:** - Ensure you have enough storage space - Try the other architecture APK - Clear cache and try again

Problem: "Unknown sources" not available **Solution:** - Look for "Install unknown apps" in newer Android versions - Check app-specific permissions in Settings

Runtime Issues

Problem: Python commands not working **Solution:** - Wait for initial setup to complete - Restart the app if needed - Check storage permissions

Problem: Pip install fails **Solution:** - Ensure internet connection - Try installing with `--user` flag - Check package name spelling

Performance Issues

Problem: App runs slowly **Solution:** - Close other apps to free RAM - Try the ARM32 version on older devices - Restart your device

📞 Support

If you encounter any issues: 1. Check this troubleshooting guide 2. Restart the app and try again 3. Ensure your device meets minimum requirements 4. Contact the developer with specific error messages

🎓 Educational Use

This app is perfect for: - **Learning Python** - Complete development environment - **Mobile Coding** - Code anywhere, anytime - **Educational Projects** - Full-featured IDE for students - **Quick Scripts** - Test ideas on the go - **Package Exploration** - Try new Python libraries

📄 License

Enjoy coding with Python Pocket IDE! 🐍📱

For the best experience, use the ARM64 version on modern devices and ensure you have adequate storage space and RAM.

APPENDIX A: COMPLETE SOURCE CODE LISTING

This section contains the complete source code for the Python Pocket IDE project, organized by file type and directory structure.

Kotlin File: app/src/main/java/com/wildzeus/pythonktx/ui/LayoutComponents/DropDownMenu/DropDown

File Path:
app/src/main/java/com/wildzeus/pythonktx/ui/LayoutComponents/DropDownMenu/DropDownMenuItem.kt

File Type: Kotlin

Lines of Code: 8

```
package com.wildzeus.pythonktx.ui.LayoutComponents.DropDownMenu

data class DropDownMenuItem(
        val Title:String,
          val Icon:Int,
     val Clickable:()->Unit
            )
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/Application.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/Application.kt

File Type: Kotlin

Lines of Code: 10

```
package com.ankit.pythonpocketide

import com.google.android.material.color.DynamicColors

class Application : android.app.Application() {
    override fun onCreate() {
        super.onCreate()
        DynamicColors.applyToActivitiesIfAvailable(this)
    }
}
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/viewModels/HomeScreenViewModel.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/viewModels/HomeScreenViewModel.kt

File Type: Kotlin

Lines of Code: 45

```
package com.ankit.pythonpocketide.viewModels

import android.app.Application
import androidx.compose.material3.DrawerState
import androidx.compose.material3.DrawerValue
import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.AndroidViewModel
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import java.io.File


class HomeScreenViewModel(application: Application) :AndroidViewModel(application) {

    private val _mDrawerState= mutableStateOf(DrawerState(DrawerValue.Closed))
                                val mDrawerState
                                 get()=_mDrawerState
                    private val _mFileName= mutableStateOf("")
                                  val mFileName
                                  get() = _mFileName
                  private val _mDialogState= mutableStateOf(false)
                                  val mDialogState
                                 get() = _mDialogState
                         fun saveFile(fileName: String){
                        CoroutineScope(Dispatchers.IO).run {
                           // create a new file in files-dir
    val fileDir = File(getApplication<Application>().filesDir.toString()+"/pythonFiles")
                         val nameWithoutSpaces=fileName.replace(" ","_")
                         val file= File(fileDir,"$nameWithoutSpaces.py")
                                   file.createNewFile()
                                      }
                                  }
                          fun changeFileName(fileName: String) {
                                  _mFileName.value=fileName
                                  }
                                 fun dismissDialog() {
                                    _mFileName.value=""
                                 _mDialogState.value=false

                                  }
                                  fun showDialog() {
                                 _mDialogState.value=true
                                  }
                                }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/models/Project.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/models/Project.kt

File Type: Kotlin

Lines of Code: 420

```
/*
 * Copyright (c) 2024 Ankit Kumar
 * Python Pocket IDE - Enhanced Mobile Python Development Environment
 *
 * Project models for project-based Python development
 */

package com.ankit.pythonpocketide.models

import java.io.File
import java.time.LocalDateTime
import java.time.format.DateTimeFormatter

/**
 * Represents a Python project with files, dependencies, and metadata
 */
data class Project(
    val name: String,
    val description: String,
    val path: String,
    val createdAt: LocalDateTime = LocalDateTime.now(),
    val modifiedAt: LocalDateTime = LocalDateTime.now(),
    val mainFile: String = "main.py",
    val dependencies: MutableList<Dependency> = mutableListOf(),
    val files: MutableList<ProjectFile> = mutableListOf(),
    val template: ProjectTemplate? = null
) {

    fun getProjectDirectory(): File = File(path)

    fun getMainFilePath(): String = "$path${File.separator}$mainFile"

    fun getRequirementsPath(): String = "$path${File.separator}requirements.txt"

    fun getPyprojectPath(): String = "$path${File.separator}pyproject.toml"

    fun getCreatedDateString(): String =
        createdAt.format(DateTimeFormatter.ofPattern("MMM dd, yyyy"))

    fun getModifiedDateString(): String =
        modifiedAt.format(DateTimeFormatter.ofPattern("MMM dd, yyyy HH:mm"))

    fun addDependency(dependency: Dependency) {
        if (!dependencies.contains(dependency)) {
            dependencies.add(dependency)
        }
    }

    fun removeDependency(packageName: String) {
        dependencies.removeIf { it.name == packageName }
    }

    fun addFile(file: ProjectFile) {
        if (!files.contains(file)) {
            files.add(file)
        }
    }

    fun removeFile(filePath: String) {
        files.removeIf { it.path == filePath }
    }

    fun getAllPythonFiles(): List<ProjectFile> =
        files.filter { it.path.endsWith(".py") }

    fun getFileCount(): Int = files.size

    fun getDependencyCount(): Int = dependencies.size
}

/**
 * Represents a Python package dependency
 */
data class Dependency(
    val name: String,
    val version: String = "",
    val isRequired: Boolean = true,
    val description: String = ""
) {
    fun getDisplayName(): String = if (version.isNotEmpty()) "$name==$version" else name

    fun toRequirementString(): String = getDisplayName()
}

/**
 * Represents a file within a project
```

```
                                    */
                                data class ProjectFile(
                                     val name: String,
                                     val path: String,
                                  val relativePath: String,
                                val isDirectory: Boolean = false,
                                       val size: Long = 0,
                            val lastModified: Long = System.currentTimeMillis()
                                    ) {
    fun getExtension(): String = if (name.contains('.')) name.substringAfterLast('.') else ""

                        fun isPythonFile(): Boolean = getExtension() == "py"

fun isConfigFile(): Boolean = name in listOf("requirements.txt", "pyproject.toml", "__init__.py")

                                fun getDisplaySize(): String {
                                          return when {
                                       size < 1024 -> "${size}B"
                                    size < 1024 * 1024 -> "${size / 1024}KB"
                                      else -> "${size / (1024 * 1024)}MB"
                                                  }
                                              }
                                             }

                                            /**
                        * Project templates for different types of Python projects
                                            */
                                enum class ProjectTemplate(
                                    val displayName: String,
                                    val description: String,
                                  val mainFileContent: String,
                           val dependencies: List<String> = emptyList(),
                       val additionalFiles: Map<String, String> = emptyMap()
                                    ) {
                                     EMPTY(
                                displayName = "Empty Project",
                             description = "A minimal Python project structure",
                          mainFileContent = """# Main entry point for your Python project
                                    print("Hello, Python Pocket IDE!")

                                          def main():
                                              '''
                                    Main function - implement your logic here
                                              '''
                                               pass

                                    if __name__ == "__main__":
                                              main()
                                            """,
                                        dependencies = emptyList()
                                                ),

                                          FLASK_API(
                                        displayName = "Flask API",
                              description = "A starter RESTful API using Flask framework",
                             mainFileContent = """from flask import Flask, jsonify, request

                                       app = Flask(__name__)

                                          # Sample data
                                            users = [
                             {"id": 1, "name": "Alice", "email": "alice@example.com"},
                                {"id": 2, "name": "Bob", "email": "bob@example.com"}
                                             ]

                                        @app.route('/')
                                          def home():
                    return jsonify({"message": "Welcome to Flask API!", "version": "1.0"})

                                @app.route('/users', methods=['GET'])
                                          def get_users():
                                         return jsonify(users)

                            @app.route('/users/<int:user_id>', methods=['GET'])
                                        def get_user(user_id):
                          user = next((u for u in users if u["id"] == user_id), None)
                                             if user:
                                          return jsonify(user)
                                return jsonify({"error": "User not found"}), 404

                                   @app.route('/users', methods=['POST'])
                                          def create_user():
                                        data = request.get_json()
                                              new_user = {
                                           "id": len(users) + 1,
                                         "name": data.get("name"),
                                        "email": data.get("email")
                                                 }
                                         users.append(new_user)
                                       return jsonify(new_user), 201

                                        if __name__ == "__main__":
                                app.run(debug=True, host='0.0.0.0', port=5000)
                                              """,
                                     dependencies = listOf("flask==2.3.3"),
                                            additionalFiles = mapOf(
```

```
                                    "api/__init__.py" to "",
                                "api/routes.py" to "# API routes module\n",
                        "config.py" to "# Configuration settings\nDEBUG = True\nPORT = 5000\n"
                                                )
                                            ),

                                    DATA_ANALYSIS(
                                displayName = "Data Analysis",
                        description = "Data analysis project with pandas and matplotlib",
                                mainFileContent = """import pandas as pd
                                 import matplotlib.pyplot as plt
                                        import numpy as np

                                    def load_sample_data():
                                '''Create sample dataset for analysis'''
                                            np.random.seed(42)
                                                 data = {
                                    'name': [f'Item_{i}' for i in range(100)],
                                     'value': np.random.normal(50, 15, 100),
                                'category': np.random.choice(['A', 'B', 'C'], 100),
                            'date': pd.date_range('2024-01-01', periods=100, freq='D')
                                                }
                                        return pd.DataFrame(data)

                                    def analyze_data(df):
                                  '''Perform basic data analysis'''
                                     print("Dataset Overview:")
                                     print(f"Shape: {df.shape}")
                                print(f"Columns: {list(df.columns)}")
                                     print("\nBasic Statistics:")
                                         print(df.describe())

                                             return df

                                  def create_visualizations(df):
                                   '''Create data visualizations'''
                                fig, axes = plt.subplots(2, 2, figsize=(12, 8))

                                        # Value distribution
                                axes[0, 0].hist(df['value'], bins=20, alpha=0.7)
                                   axes[0, 0].set_title('Value Distribution')
                                         axes[0, 0].set_xlabel('Value')
                                        axes[0, 0].set_ylabel('Frequency')

                                        # Category counts
                                df['category'].value_counts().plot(kind='bar', ax=axes[0, 1])
                                        axes[0, 1].set_title('Category Distribution')
                                              axes[0, 1].set_xlabel('Category')
                                                axes[0, 1].set_ylabel('Count')

                                        # Time series
                                    daily_avg = df.groupby('date')['value'].mean()
                                  axes[1, 0].plot(daily_avg.index, daily_avg.values)
                                     axes[1, 0].set_title('Daily Average Values')
                                            axes[1, 0].set_xlabel('Date')
                                       axes[1, 0].set_ylabel('Average Value')

                                        # Box plot by category
                                     categories = df['category'].unique()
                    category_data = [df[df['category'] == cat]['value'] for cat in categories]
                               axes[1, 1].boxplot(category_data, labels=categories)
                              axes[1, 1].set_title('Value Distribution by Category')
                                        axes[1, 1].set_xlabel('Category')
                                          axes[1, 1].set_ylabel('Value')

                                        plt.tight_layout()
                        plt.savefig('analysis_results.png', dpi=300, bbox_inches='tight')
                             print("Visualizations saved as 'analysis_results.png'")

                                    def main():
                                 '''Main analysis workflow'''
                                print("Starting Data Analysis...")

                                    # Load and analyze data
                                    df = load_sample_data()
                                     df = analyze_data(df)

                                    # Create visualizations
                                   create_visualizations(df)

                                print("\\nAnalysis complete!")

                                if __name__ == "__main__":
                                            main()
                                           """,
                    dependencies = listOf("pandas==2.1.3", "matplotlib==3.8.2", "numpy==1.25.2"),
                                    additionalFiles = mapOf(
                                      "data/__init__.py" to "",
                                "data/loader.py" to "# Data loading utilities\n",
                                        "analysis/__init__.py" to "",
                            "analysis/statistics.py" to "# Statistical analysis functions\n"
                                                )
                                            ),

                                    MACHINE_LEARNING(
                                displayName = "Machine Learning",
```

```
        description = "ML project template with scikit-learn",
                mainFileContent = """import numpy as np
                        import pandas as pd
    from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
     from sklearn.datasets import make_classification
                            import joblib

                    def create_sample_dataset():
                '''Generate sample classification dataset'''
                        X, y = make_classification(
                                n_samples=1000,
                                n_features=10,
                               n_informative=5,
                                n_redundant=3,
                                 n_classes=3,
                                random_state=42
                                     )

    feature_names = [f'feature_{i}' for i in range(X.shape[1])]
            df = pd.DataFrame(X, columns=feature_names)
                            df['target'] = y

                                return df

                        def preprocess_data(df):
                        '''Preprocess the dataset'''
                       print("Preprocessing data...")

                       # Separate features and target
                        X = df.drop('target', axis=1)
                               y = df['target']

                      # Split into train and test sets
          X_train, X_test, y_train, y_test = train_test_split(
              X, y, test_size=0.2, random_state=42, stratify=y
                                     )

         print(f"Training set: {X_train.shape[0]} samples")
            print(f"Test set: {X_test.shape[0]} samples")

                   return X_train, X_test, y_train, y_test

                    def train_model(X_train, y_train):
                    '''Train a Random Forest classifier'''
                   print("Training Random Forest model...")

                       model = RandomForestClassifier(
                                n_estimators=100,
                                  max_depth=10,
                                 random_state=42
                                     )

                         model.fit(X_train, y_train)

                       print("Model training completed!")
                                 return model

                def evaluate_model(model, X_test, y_test):
                        '''Evaluate the trained model'''
                          print("Evaluating model...")

                               # Make predictions
                         y_pred = model.predict(X_test)

                              # Calculate accuracy
                   accuracy = accuracy_score(y_test, y_pred)
                      print(f"Accuracy: {accuracy:.4f}")

                       # Detailed classification report
                report = classification_report(y_test, y_pred)
                        print("Classification Report:")
                                 print(report)

                                return accuracy

               def save_model(model, filename='model.pkl'):
                           '''Save the trained model'''
                           joblib.dump(model, filename)
                     print(f"Model saved as '{filename}'")

                                def main():
                              '''Main ML workflow'''
                print("Starting Machine Learning Pipeline...")

                             # Create and preprocess data
                             df = create_sample_dataset()
          X_train, X_test, y_train, y_test = preprocess_data(df)

                              # Train and evaluate model
                       model = train_model(X_train, y_train)
           accuracy = evaluate_model(model, X_test, y_test)

                                # Save the model
                                save_model(model)
```

```
print(f"\\nML Pipeline completed! Final accuracy: {accuracy:.4f}")

if __name__ == "__main__":
    main()
""",
    dependencies = listOf(
      "scikit-learn==1.3.2",
        "pandas==2.1.3",
        "numpy==1.25.2",
        "joblib==1.3.2"
        ),
    additionalFiles = mapOf(
      "models/__init__.py" to "",
  "models/train.py" to "# Model training utilities\n",
"models/evaluate.py" to "# Model evaluation utilities\n",
      "data/__init__.py" to "",
      "utils/__init__.py" to "",
"utils/preprocessing.py" to "# Data preprocessing utilities\n"
      )
    );

fun getInitFileContent(): String = "# ${displayName} - Generated by Python Pocket IDE\n"

fun getPyprojectContent(projectName: String): String = """[build-system]
  requires = ["setuptools>=45", "wheel"]
build-backend = "setuptools.build_meta"

    [project]
  name = "$projectName"
    version = "0.1.0"
description = "$description"
    authors = [
  {name = "Python Pocket IDE User", email = "user@example.com"},
    ]
    dependencies = [
  ${dependencies.joinToString(",\n") { "    \"$it\"" }}
    ]
  requires-python = ">=3.8"

    [project.scripts]
$projectName = "$projectName.main:main"
    """

fun getRequirementsContent(): String = dependencies.joinToString("\n")
    }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/dataStore/SettingsDataStore.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/dataStore/SettingsDataStore.kt

File Type: Kotlin

Lines of Code: 39

```
package com.ankit.pythonpocketide.dataStore

import android.content.Context
import androidx.datastore.preferences.core.booleanPreferencesKey
import androidx.datastore.preferences.core.edit
import androidx.datastore.preferences.core.stringPreferencesKey
import androidx.datastore.preferences.preferencesDataStore
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.map

class SettingsDataStore(private val mContext: Context) {
                companion object{
            val Context.dataStore by preferencesDataStore(
                    name = "Settings"
                    )
                }

                private object PreferencesKeys {
                val Theme = stringPreferencesKey("Theme")
            val areFilesExtracted= booleanPreferencesKey("FilesStatus")
                }
    val mThemeString: Flow<String?> = mContext.dataStore.data.map { preferences ->
                    preferences[PreferencesKeys.Theme]
                }
val  areFilesExtracted: Flow<Boolean?> = mContext.dataStore.data.map { preferences ->
                preferences[PreferencesKeys.areFilesExtracted]
                }
                suspend fun updateTheme(mTheme: String) {
                   mContext.dataStore.edit { preferences ->
                    preferences[PreferencesKeys.Theme] = mTheme
                    }
                }
            suspend fun updateFileStatus(fileStatus:Boolean) {
                   mContext.dataStore.edit { preferences ->
                preferences[PreferencesKeys.areFilesExtracted] = fileStatus
                    }
                }
            }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/utils/ProjectManager.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/utils/ProjectManager.kt

File Type: Kotlin

Lines of Code: 401

```
/*
 * Copyright (c) 2024 Ankit Kumar
 * Python Pocket IDE - Enhanced Mobile Python Development Environment
 *
 * ProjectManager handles all project-related operations
 */

package com.ankit.pythonpocketide.utils

import android.content.Context
import android.util.Log
import com.ankit.pythonpocketide.models.Dependency
import com.ankit.pythonpocketide.models.Project
import com.ankit.pythonpocketide.models.ProjectFile
import com.ankit.pythonpocketide.models.ProjectTemplate
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.withContext
import java.io.File
import java.io.FileInputStream
import java.io.FileOutputStream
import java.io.IOException
import java.time.LocalDateTime
import java.util.zip.ZipEntry
import java.util.zip.ZipInputStream
import java.util.zip.ZipOutputStream

class ProjectManager private constructor(private val context: Context) {

    companion object {
        private const val TAG = "ProjectManager"
        private const val PROJECTS_DIR = "projects"
        private const val LEGACY_FILES_DIR = "pythonFiles"

        @Volatile
        private var INSTANCE: ProjectManager? = null

        fun getInstance(context: Context): ProjectManager {
            return INSTANCE ?: synchronized(this) {
                INSTANCE ?: ProjectManager(context.applicationContext).also { INSTANCE = it }
            }
        }
    }

    private val projectsDirectory: File by lazy {
        File(context.filesDir, PROJECTS_DIR).apply {
            if (!exists()) mkdirs()
        }
    }

    private val legacyFilesDirectory: File by lazy {
        File(context.filesDir, LEGACY_FILES_DIR).apply {
            if (!exists()) mkdirs()
        }
    }

    /**
     * Get all projects
     */
    suspend fun getAllProjects(): List<Project> = withContext(Dispatchers.IO) {
        try {
            projectsDirectory.listFiles()
                ?.filter { it.isDirectory }
                ?.mapNotNull { loadProject(it.name) }
                ?.sortedByDescending { it.modifiedAt }
                ?: emptyList()
        } catch (e: Exception) {
            Log.e(TAG, "Error loading projects", e)
            emptyList()
        }
    }

    /**
     * Create a new project from template
     */
    suspend fun createProject(
        name: String,
        description: String,
        template: ProjectTemplate
    ): Result<Project> = withContext(Dispatchers.IO) {
        try {
            // Validate project name
            if (!isValidProjectName(name)) {
                return@withContext Result.failure(IllegalArgumentException("Invalid project name"))
            }
```

```
                            val projectDir = File(projectsDirectory, name)
                                    if (projectDir.exists()) {
        return@withContext Result.failure(IllegalArgumentException("Project already exists"))
                                            }

                                    // Create project directory
                                        projectDir.mkdirs()

                                    // Create project structure
                                       val project = Project(
                                              name = name,
                                       description = description,
                                      path = projectDir.absolutePath,
                                            template = template
                                                    )

                                        // Create main file
                                    val mainFile = File(projectDir, "main.py")
                                   mainFile.writeText(template.mainFileContent)

                                        // Create __init__.py
                                    val initFile = File(projectDir, "__init__.py")
                                  initFile.writeText(template.getInitFileContent())

                                    // Create requirements.txt
                                      if (template.dependencies.isNotEmpty()) {
                                 val requirementsFile = File(projectDir, "requirements.txt")
                                requirementsFile.writeText(template.getRequirementsContent())
                                                    }

                                   // Create pyproject.toml
                                val pyprojectFile = File(projectDir, "pyproject.toml")
                               pyprojectFile.writeText(template.getPyprojectContent(name))

                                      // Create additional files and directories
                                 template.additionalFiles.forEach { (path, content) ->
                                             val file = File(projectDir, path)
                                                 file.parentFile?.mkdirs()
                                                  file.writeText(content)
                                                    }

                                    // Scan and add files to project
                                   val updatedProject = project.copy(
                                    files = scanProjectFiles(projectDir),
                        dependencies = parseDependencies(File(projectDir, "requirements.txt"))
                                                    )

                                     Result.success(updatedProject)
                                      } catch (e: Exception) {
                                 Log.e(TAG, "Error creating project", e)
                                            Result.failure(e)
                                                  }
                                              }

                                             /**
                                    * Load an existing project
                                              */
suspend fun loadProject(projectName: String): Project? = withContext(Dispatchers.IO) {
                                              try {
                                 val projectDir = File(projectsDirectory, projectName)
                                if (!projectDir.exists() || !projectDir.isDirectory) {
                                             return@withContext null
                                                    }

                                 val pyprojectFile = File(projectDir, "pyproject.toml")
                                     val description = if (pyprojectFile.exists()) {
                                parsePyprojectDescription(pyprojectFile) ?: "Python project"
                                                } else {
                                            "Python project"
                                                  }

                                                Project(
                                              name = projectName,
                                           description = description,
                                          path = projectDir.absolutePath,
                                    createdAt = LocalDateTime.ofEpochSecond(
                                       projectDir.lastModified() / 1000, 0,
                        java.time.ZoneOffset.systemDefault().rules.getOffset(LocalDateTime.now())
                                                    ),
                                   modifiedAt = LocalDateTime.ofEpochSecond(
                                        projectDir.lastModified() / 1000, 0,
                        java.time.ZoneOffset.systemDefault().rules.getOffset(LocalDateTime.now())
                                                    ),
                                    files = scanProjectFiles(projectDir),
                        dependencies = parseDependencies(File(projectDir, "requirements.txt"))
                                                    )
                                    } catch (e: Exception) {
                               Log.e(TAG, "Error loading project: $projectName", e)
                                                null
                                                  }
                                              }

                                             /**
                                      * Delete a project
                                              */
```

```
    suspend fun deleteProject(projectName: String): Result<Unit> = withContext(Dispatchers.IO) {
                                        try {
                            val projectDir = File(projectsDirectory, projectName)
                                    if (projectDir.exists()) {
                                    projectDir.deleteRecursively()
                                                }
                                       Result.success(Unit)
                                   } catch (e: Exception) {
                                Log.e(TAG, "Error deleting project", e)
                                        Result.failure(e)
                                              }
                                            }

                                        /**
                            * Convert a standalone Python file to a project
                                        */
                                  suspend fun convertFileToProject(
                                            filePath: String,
                                           projectName: String,
                                            description: String
                           ): Result<Project> = withContext(Dispatchers.IO) {
                                        try {
                                    val sourceFile = File(filePath)
                                      if (!sourceFile.exists()) {
                return@withContext Result.failure(IllegalArgumentException("Source file not found"))
                                                }

                                       // Read original file content
                                    val fileContent = sourceFile.readText()

                                     // Create project using empty template
                      val result = createProject(projectName, description, ProjectTemplate.EMPTY)

                                          if (result.isSuccess) {
                                        val project = result.getOrThrow()
                                // Replace main.py content with original file content
                                     val mainFile = File(project.path, "main.py")
                                            mainFile.writeText(fileContent)

                                              Result.success(project)
                                                   } else {
                                                      result
                                                    }
                                           } catch (e: Exception) {
                                  Log.e(TAG, "Error converting file to project", e)
                                               Result.failure(e)
                                                     }
                                                   }

                                               /**
                                         * Export project as ZIP
                                                */
suspend fun exportProject(project: Project, outputFile: File): Result<Unit> = withContext(Dispatchers
                                                try {
                               ZipOutputStream(FileOutputStream(outputFile)).use { zipOut ->
                                          val projectDir = File(project.path)
                                         zipDirectory(projectDir, "", zipOut)
                                                        }
                                               Result.success(Unit)
                                             } catch (e: Exception) {
                                        Log.e(TAG, "Error exporting project", e)
                                                Result.failure(e)
                                                      }
                                                    }

                                                /**
                                         * Import project from ZIP
                                                */
suspend fun importProject(zipFile: File, projectName: String): Result<Project> = withContext(Dispatch
                                                try {
                                    val projectDir = File(projectsDirectory, projectName)
                                                if (projectDir.exists()) {
                    return@withContext Result.failure(IllegalArgumentException("Project already exists"))
                                                        }

                                                projectDir.mkdirs()

                                ZipInputStream(FileInputStream(zipFile)).use { zipIn ->
                                                  var entry = zipIn.nextEntry
                                                    while (entry != null) {
                                           val file = File(projectDir, entry.name)
                                                  if (entry.isDirectory) {
                                                          file.mkdirs()
                                                           } else {
                                                     file.parentFile?.mkdirs()
                                                FileOutputStream(file).use { output ->
                                                           zipIn.copyTo(output)
                                                                  }
                                                               }
                                                    entry = zipIn.nextEntry
                                                      }
                                                    }

                                               // Load the imported project
                                             loadProject(projectName)?.let { project ->
                                                Result.success(project)
```

```
                        } ?: Result.failure(Exception("Failed to load imported project"))
                                    } catch (e: Exception) {
                                Log.e(TAG, "Error importing project", e)
                                           Result.failure(e)
                                                 }
                                             }

                                            /**
                         * Get legacy Python files (for backward compatibility)
                                             */
                 suspend fun getLegacyFiles(): List<File> = withContext(Dispatchers.IO) {
                                             try {
                                      legacyFilesDirectory.listFiles()
                                   ?.filter { it.isFile && it.extension == "py" }
                                               ?.toList()
                                            ?: emptyList()
                                    } catch (e: Exception) {
                                Log.e(TAG, "Error loading legacy files", e)
                                           emptyList()
                                              }
                                           }

                                            /**
                                       * Install project dependencies
                                             */
  suspend fun installDependencies(project: Project): Result<String> = withContext(Dispatchers.IO) {
                                             try {
                          val requirementsFile = File(project.path, "requirements.txt")
                                       if (!requirementsFile.exists()) {
                           return@withContext Result.success("No dependencies to install")
                                                }

         val command = Commands.getEnhancedPipInstallCommand(context, "-r ${requirementsFile.absoluteP
                                          Result.success(command)
                                       } catch (e: Exception) {
                                Log.e(TAG, "Error installing dependencies", e)
                                           Result.failure(e)
                                                 }
                                             }

                                       // Private helper methods

                        private fun isValidProjectName(name: String): Boolean {
                                      return name.isNotBlank() &&
                                   name.matches(Regex("[a-zA-Z0-9_-]+")) &&
                                               name.length <= 50
                                                }

              private fun scanProjectFiles(projectDir: File): MutableList<ProjectFile> {
                                  val files = mutableListOf<ProjectFile>()

                        fun scanDirectory(dir: File, relativePath: String = "") {
                                      dir.listFiles()?.forEach { file ->
               val relPath = if (relativePath.isEmpty()) file.name else "$relativePath/${file.name}"
                                           files.add(
                                            ProjectFile(
                                            name = file.name,
                                          path = file.absolutePath,
                                           relativePath = relPath,
                                        isDirectory = file.isDirectory,
                                    size = if (file.isFile) file.length() else 0,
                                         lastModified = file.lastModified()
                                                    )
                                                )

                                           if (file.isDirectory) {
                                          scanDirectory(file, relPath)
                                                    }
                                                }
                                             }

                                       scanDirectory(projectDir)
                                             return files
                                                }

          private fun parseDependencies(requirementsFile: File): MutableList<Dependency> {
                        if (!requirementsFile.exists()) return mutableListOf()

                                  return requirementsFile.readLines()
                              .filter { it.isNotBlank() && !it.startsWith("#") }
                                          .mapNotNull { line ->
                                          val trimmed = line.trim()
                                                   when {
                                              "==" in trimmed -> {
                                           val parts = trimmed.split("==")
                                                if (parts.size == 2) {
                                             Dependency(parts[0], parts[1])
                                                    } else null
                                                       }
                                           else -> Dependency(trimmed)
                                                     }
                                                }
                                            .toMutableList()
                                                }

                        private fun parsePyprojectDescription(file: File): String? {
```

```
                                            return try {
                                            file.readLines()
                                    .find { it.trim().startsWith("description") }
                                            ?.substringAfter("=")
                                                  ?.trim()
                                           ?.removeSurrounding("\"")
                                         } catch (e: Exception) {
                                                    null
                                                   }
                                                }

private fun zipDirectory(dir: File, parentPath: String, zipOut: ZipOutputStream) {
                                dir.listFiles()?.forEach { file ->
    val entryPath = if (parentPath.isEmpty()) file.name else "$parentPath/${file.name}"

                                        if (file.isDirectory) {
                                    zipOut.putNextEntry(ZipEntry("$entryPath/"))
                                                zipOut.closeEntry()
                                       zipDirectory(file, entryPath, zipOut)
                                                   } else {
                                      zipOut.putNextEntry(ZipEntry(entryPath))
                                        FileInputStream(file).use { input ->
                                                  input.copyTo(zipOut)
                                                       }
                                                zipOut.closeEntry()
                                                      }
                                                   }
                                                }
                                                }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/utils/PermissionManageExternal.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/utils/PermissionManageExternal.kt

File Type: Kotlin

Lines of Code: 28

```
package com.ankit.pythonpocketide.utils

import android.app.Activity
import android.content.Intent
import android.net.Uri
import android.provider.Settings
import androidx.annotation.RequiresApi

@RequiresApi(30)
object PermissionManageExternal {
fun request(activity: Activity): Boolean {
try {
val uri = Uri.parse("package:" + com.ankit.pythonpocketide.BuildConfig.APPLICATION_ID)
val intent = Intent(Settings.ACTION_MANAGE_APP_ALL_FILES_ACCESS_PERMISSION, uri)
activity.startActivity(intent)
return true
} catch (ignore: Exception) {
}
try {
val intent = Intent()
intent.action = Settings.ACTION_MANAGE_ALL_FILES_ACCESS_PERMISSION
activity.startActivity(intent)
return true
} catch (ignore: Exception) {
}
return false
}
}
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/utils/PythonFileManager.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/utils/PythonFileManager.kt

File Type: Kotlin

Lines of Code: 42

```
package com.ankit.pythonpocketide.utils
import android.os.Build
import android.os.FileObserver
import androidx.compose.runtime.mutableStateOf
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import java.io.File

object PythonFileManager {
var  pythonFiles = mutableStateOf( listOf<File>())
private lateinit var observer: FileObserver
var filesDir: String = ""
fun init() {
pythonFiles.value = File(filesDir).listFiles { _, name -> name!!.endsWith(".py") }?.toList() ?: l
observer = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
val file = File(filesDir)
object : FileObserver(file) {
override fun onEvent(event: Int, file: String?) {
if (event == CREATE || event == DELETE || event == MODIFY) {
updatePythonFileList()
}
}
}
} else {
object : FileObserver(filesDir) {
override fun onEvent(event: Int, file: String?) {
if (event == CREATE || event == DELETE || event == MODIFY) {
updatePythonFileList()
}
}
}
}
observer.startWatching()
}
fun updatePythonFileList(){
CoroutineScope(Dispatchers.Main).run {
pythonFiles.value = (File(filesDir).listFiles { _, name -> name!!.endsWith(".py") }?.toList()
it.lastModified()
}.reversed()
}
}
}
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/utils/Commands.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/utils/Commands.kt

File Type: Kotlin

Lines of Code: 213

```
package com.ankit.pythonpocketide.utils

import android.content.Context
import java.io.File

object Commands {
    fun getBasicCommand(context: Context): String {
        val appLibDirPath = context.applicationInfo.nativeLibraryDir
        val appFileDirPath = context.filesDir.absolutePath
        val pythonBuildDirPath = "$appFileDirPath/files/usr"
        val pythonLibDirPath = "$pythonBuildDirPath/lib"
        val pythonExecName = "libpython3.so"
 val aliasCommand = "alias python=\"$pythonExecName\" && alias pip=\"$pythonExecName -m pip \""
return "export PATH=\$PATH:$appLibDirPath && export PYTHONHOME=$pythonBuildDirPath && export PYTH
    }
    fun getInterpreterCommand(context: Context, filePath: String): String {
        val appLibDirPath = context.applicationInfo.nativeLibraryDir
        val appFileDirPath = context.filesDir.absolutePath
        val pythonBuildDirPath = "$appFileDirPath/files/usr"
        val pythonLibDirPath = "$pythonBuildDirPath/lib"
        val pythonExecName = "libpython3.so"
return "export PATH=\$PATH:$appLibDirPath && export PYTHONHOME=$pythonBuildDirPath && export LD_L
    }

    fun getPythonShellCommand(context: Context): String {
        val appLibDirPath = context.applicationInfo.nativeLibraryDir
        val appFileDirPath = context.filesDir.absolutePath
        val pythonBuildDirPath = "$appFileDirPath/files/usr"
        val pythonLibDirPath = "$pythonBuildDirPath/lib"
return "export PATH=\$PATH:$appLibDirPath && export PYTHONHOME=$pythonBuildDirPath && export PYTH
    }

    // Enhanced commands for better functionality
    fun getEnhancedBasicCommand(context: Context): String {
        val appLibDirPath = context.applicationInfo.nativeLibraryDir
        val appFileDirPath = context.filesDir.absolutePath
        val pythonBuildDirPath = "$appFileDirPath/files/usr"
        val pythonLibDirPath = "$pythonBuildDirPath/lib"
        val pythonExecName = "libpython3.so"
val aliasCommand = "alias python=\"$pythonExecName\" && alias pip=\"$pythonExecName -m pip\" && a
val envSetup = "export PATH=\$PATH:$appLibDirPath && export PYTHONHOME=$pythonBuildDirPath && exp
return "$envSetup && $aliasCommand && clear && echo 'Python Pocket IDE Terminal' && echo 'Python
    }

    fun getEnhancedInterpreterCommand(context: Context, filePath: String): String {
        val appLibDirPath = context.applicationInfo.nativeLibraryDir
        val appFileDirPath = context.filesDir.absolutePath
        val pythonBuildDirPath = "$appFileDirPath/files/usr"
        val pythonLibDirPath = "$pythonBuildDirPath/lib"
        val pythonExecName = "libpython3.so"
val envSetup = "export PATH=\$PATH:$appLibDirPath && export PYTHONHOME=$pythonBuildDirPath && exp
return "$envSetup && clear && echo 'Running: $filePath' && $pythonExecName $filePath && echo '' &
    }

    fun getEnhancedPythonShellCommand(context: Context): String {
        val appLibDirPath = context.applicationInfo.nativeLibraryDir
        val appFileDirPath = context.filesDir.absolutePath
        val pythonBuildDirPath = "$appFileDirPath/files/usr"
        val pythonLibDirPath = "$pythonBuildDirPath/lib"
        val pythonExecName = "libpython3.so"
val envSetup = "export PATH=\$PATH:$appLibDirPath && export PYTHONHOME=$pythonBuildDirPath && exp
return "$envSetup && clear && echo 'Python 3.11.5 Interactive Shell' && echo 'Type exit() to quit
    }

    fun getEnhancedPipInstallCommand(context: Context, packageName: String): String {
        val appLibDirPath = context.applicationInfo.nativeLibraryDir
        val appFileDirPath = context.filesDir.absolutePath
        val pythonBuildDirPath = "$appFileDirPath/files/usr"
        val pythonLibDirPath = "$pythonBuildDirPath/lib"
        val pythonExecName = "libpython3.so"
val envSetup = "export PATH=\$PATH:$appLibDirPath && export PYTHONHOME=$pythonBuildDirPath && exp
return "$envSetup && clear && echo 'Installing package: $packageName' && $pythonExecName -m pip i
    }

    // PROJECT-AWARE COMMANDS FOR PROJECT-BASED DEVELOPMENT

    /**
     * Get command to run a project file with proper project context
     */
fun getProjectFileCommand(context: Context, projectPath: String, filePath: String): String {
        val appLibDirPath = context.applicationInfo.nativeLibraryDir
        val appFileDirPath = context.filesDir.absolutePath
        val pythonBuildDirPath = "$appFileDirPath/files/usr"
        val pythonLibDirPath = "$pythonBuildDirPath/lib"
        val pythonExecName = "libpython3.so"
```

```
                    // Set up Python path to include project directory for imports
                        val envSetup = "export PATH=\$PATH:$appLibDirPath && " +
                                       "export PYTHONHOME=$pythonBuildDirPath && " +
                    "export PYTHONPATH=\"$projectPath:$appLibDirPath:$pythonLibDirPath\" && " +
                         "export LD_LIBRARY_PATH=\"\$LD_LIBRARY_PATH:$pythonLibDirPath\""

                        return "$envSetup && cd \"$projectPath\" && clear && " +
            "echo 'Running: ${File(filePath).name} (Project: ${File(projectPath).name})' && " +
                                "$pythonExecName \"$filePath\" && echo '' && " +
                    "echo '[Execution completed - Enter to exit]' && read junk && exit"
                                                }

                                                /**
                                             * Get command to run project main file
                                                */
fun getProjectMainCommand(context: Context, projectPath: String, mainFile: String = "main.py"): Strin
                        val mainFilePath = "$projectPath${File.separator}$mainFile"
                    return getProjectFileCommand(context, projectPath, mainFilePath)
                                                }

                                                /**
                                * Get command to start Python shell in project context
                                                */
            fun getProjectShellCommand(context: Context, projectPath: String): String {
                        val appLibDirPath = context.applicationInfo.nativeLibraryDir
                             val appFileDirPath = context.filesDir.absolutePath
                            val pythonBuildDirPath = "$appFileDirPath/files/usr"
                              val pythonLibDirPath = "$pythonBuildDirPath/lib"
                                    val pythonExecName = "libpython3.so"

                        val envSetup = "export PATH=\$PATH:$appLibDirPath && " +
                                       "export PYTHONHOME=$pythonBuildDirPath && " +
                    "export PYTHONPATH=\"$projectPath:$appLibDirPath:$pythonLibDirPath\" && " +
                         "export LD_LIBRARY_PATH=\"\$LD_LIBRARY_PATH:$pythonLibDirPath\""

                        return "$envSetup && cd \"$projectPath\" && clear && " +
                    "echo 'Python 3.11.5 Shell - Project: ${File(projectPath).name}' && " +
                            "echo 'Project modules are available for import' && " +
                       "$pythonExecName && echo '[Enter to Exit]' && read junk && exit"
                                                }

                                                /**
                                    * Get command to install project dependencies
                                                */
      fun getProjectDependencyInstallCommand(context: Context, projectPath: String): String {
                        val appLibDirPath = context.applicationInfo.nativeLibraryDir
                             val appFileDirPath = context.filesDir.absolutePath
                            val pythonBuildDirPath = "$appFileDirPath/files/usr"
                              val pythonLibDirPath = "$pythonBuildDirPath/lib"
                                    val pythonExecName = "libpython3.so"
                 val requirementsFile = "$projectPath${File.separator}requirements.txt"

                        val envSetup = "export PATH=\$PATH:$appLibDirPath && " +
                                       "export PYTHONHOME=$pythonBuildDirPath && " +
                    "export PYTHONPATH=\"$projectPath:$appLibDirPath:$pythonLibDirPath\" && " +
                         "export LD_LIBRARY_PATH=\"\$LD_LIBRARY_PATH:$pythonLibDirPath\""

                        return "$envSetup && cd \"$projectPath\" && clear && " +
                "echo 'Installing dependencies for project: ${File(projectPath).name}' && " +
                                    "if [ -f \"requirements.txt\" ]; then " +
                               "$pythonExecName -m pip install --user -r requirements.txt; " +
                                     "else echo 'No requirements.txt found'; fi && " +
             "echo '' && echo '[Installation completed - Enter to exit]' && read junk && exit"
                                                }

                                                /**
                                * Get command for project terminal with all project context
                                                */
          fun getProjectTerminalCommand(context: Context, projectPath: String): String {
                        val appLibDirPath = context.applicationInfo.nativeLibraryDir
                             val appFileDirPath = context.filesDir.absolutePath
                            val pythonBuildDirPath = "$appFileDirPath/files/usr"
                              val pythonLibDirPath = "$pythonBuildDirPath/lib"
                                    val pythonExecName = "libpython3.so"

                        val aliasCommand = "alias python=\"$pythonExecName\" && " +
                                           "alias pip=\"$pythonExecName -m pip\" && " +
                                                "alias python3=\"$pythonExecName\""

                        val envSetup = "export PATH=\$PATH:$appLibDirPath && " +
                                       "export PYTHONHOME=$pythonBuildDirPath && " +
                    "export PYTHONPATH=\"$projectPath:$appLibDirPath:$pythonLibDirPath\" && " +
                         "export LD_LIBRARY_PATH=\"\$LD_LIBRARY_PATH:$pythonLibDirPath\""

                return "$envSetup && $aliasCommand && cd \"$projectPath\" && clear && " +
                                    "echo 'Python Pocket IDE - Project Terminal' && " +
                                     "echo 'Project: ${File(projectPath).name}' && " +
                                      "echo 'Working directory: $projectPath' && " +
                            "echo 'Python 3.11.5 ready - project modules available for import'"
                                                }

                                                /**
                                * Get command to run a specific Python module within a project
                                                */
  fun getProjectModuleCommand(context: Context, projectPath: String, moduleName: String): String {
                        val appLibDirPath = context.applicationInfo.nativeLibraryDir
```

```
                val appFileDirPath = context.filesDir.absolutePath
                val pythonBuildDirPath = "$appFileDirPath/files/usr"
                  val pythonLibDirPath = "$pythonBuildDirPath/lib"
                        val pythonExecName = "libpython3.so"

                val envSetup = "export PATH=\$PATH:$appLibDirPath && " +
                            "export PYTHONHOME=$pythonBuildDirPath && " +
              "export PYTHONPATH=\"$projectPath:$appLibDirPath:$pythonLibDirPath\" && " +
                   "export LD_LIBRARY_PATH=\"\$LD_LIBRARY_PATH:$pythonLibDirPath\""

                return "$envSetup && cd \"$projectPath\" && clear && " +
          "echo 'Running module: $moduleName (Project: ${File(projectPath).name})' && " +
                      "$pythonExecName -m $moduleName && echo '' && " +
                "echo '[Execution completed - Enter to exit]' && read junk && exit"
                                        }

                                        /**
         * Get environment setup string for project context (helper function)
                                         */
private fun getProjectEnvironmentSetup(context: Context, projectPath: String): String {
                val appLibDirPath = context.applicationInfo.nativeLibraryDir
                     val appFileDirPath = context.filesDir.absolutePath
                    val pythonBuildDirPath = "$appFileDirPath/files/usr"
                      val pythonLibDirPath = "$pythonBuildDirPath/lib"

                      return "export PATH=\$PATH:$appLibDirPath && " +
                           "export PYTHONHOME=$pythonBuildDirPath && " +
            "export PYTHONPATH=\"$projectPath:$appLibDirPath:$pythonLibDirPath\" && " +
                 "export LD_LIBRARY_PATH=\"\$LD_LIBRARY_PATH:$pythonLibDirPath\""
                                        }
                                       }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/utils/Keys.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/utils/Keys.kt

File Type: Kotlin

Lines of Code: 15

```
package com.ankit.pythonpocketide.utils

object Keys {
    const val KEY_FILE_PATH = "file_path"
const val IS_SHELL_MODE_KEY = "is_shell_mode"

// Additional keys for enhanced functionality
 const val KEY_PROJECT_PATH = "project_path"
  const val KEY_PROJECT_RUN = "project_run"
  const val KEY_PIP_INSTALL = "pip_install"
  const val KEY_PIP_PACKAGE = "pip_package"
const val KEY_TERMINAL_MODE = "terminal_mode"
 const val KEY_PYTHON_SHELL = "python_shell"
const val KEY_ENHANCED_MODE = "enhanced_mode"
}
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/activities/EditorActivity.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/activities/EditorActivity.kt

File Type: Kotlin

Lines of Code: 455

```
package com.ankit.pythonpocketide.activities

import android.content.DialogInterface
import android.graphics.Typeface
import android.graphics.drawable.Icon
import android.os.Bundle
import android.system.ErrnoException
import android.view.Menu
import android.view.MenuItem
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.blankj.utilcode.util.UriUtils
import com.google.android.material.dialog.MaterialAlertDialogBuilder
import com.ankit.pythonpocketide.R
import com.ankit.pythonpocketide.dataStore.SettingsDataStore
import com.ankit.pythonpocketide.databinding.ActivityEditorBinding
import com.ankit.pythonpocketide.ui.theme.EditorTheme
import com.ankit.pythonpocketide.utils.Keys
import com.ankit.pythonpocketide.utils.PythonFileManager
import io.github.rosemoe.sora.event.ContentChangeEvent
import io.github.rosemoe.sora.event.EditorKeyEvent
import io.github.rosemoe.sora.event.KeyBindingEvent
import io.github.rosemoe.sora.event.SelectionChangeEvent
import io.github.rosemoe.sora.langs.textmate.TextMateColorScheme
import io.github.rosemoe.sora.langs.textmate.TextMateLanguage
import io.github.rosemoe.sora.text.LineSeparator
import io.github.rosemoe.sora.widget.CodeEditor
import io.github.rosemoe.sora.widget.style.builtin.ScaleCursorAnimator
import io.github.rosemoe.sora.widget.subscribeEvent
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.flow.first
import kotlinx.coroutines.launch
import org.eclipse.tm4e.core.registry.IGrammarSource
import org.eclipse.tm4e.core.registry.IThemeSource
import java.io.File
import java.io.FileNotFoundException
import java.io.FileOutputStream
import java.io.InputStreamReader


class EditorActivity : AppCompatActivity() {
private lateinit var binding: ActivityEditorBinding
private var undo: MenuItem? = null
private var redo: MenuItem? = null
private lateinit var currentFile: File
private lateinit var dataStore: SettingsDataStore
private var currentFiles: List<File> = emptyList()

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
com.ankit.pythonpocketide.utils.CrashHandler.INSTANCE.init(this)
binding = ActivityEditorBinding.inflate(layoutInflater)
setContentView(binding.root)
binding.symbolInput.bindEditor(binding.editor)
setCurrentFile()
dataStore = SettingsDataStore(applicationContext)
binding.materialToolbar.setTitleTextAppearance(this,R.style.RobotoBoldTextAppearance)
binding.runCode.setImageIcon(Icon.createWithResource(this,R.drawable.code_run_icon))
setSupportActionBar(binding.materialToolbar)
binding.symbolInput.addSymbols(
arrayOf(
"->",
"{",
"}",
"(",
")",
",",
".",
";",
"\"",
"?",
"+",
"-",
"*",
"/",
"!",
"&",
"#",
"=",
"-"
),
arrayOf(
```

```
                                        "\t",
                                        "{}",
                                        "}",
                                        "(",
                                        ")",
                                        ",",
                                        ".",
                                        ";",
                                        "\"",
                                        "?",
                                        "+",
                                        "-",
                                        "*",
                                        "/",
                                        "!",
                                        "&",
                                        "#",
                                        "=",
                                        "-"
                                    )
                                )
    val typeface = Typeface.createFromAsset(assets, "JetBrainsMono-Regular.ttf")
                    if (PythonFileManager.filesDir.isEmpty()) {
                    PythonFileManager.filesDir = filesDir.absolutePath
                                PythonFileManager.init()
                                }
                            binding.symbolInput.forEachButton {
                                    it.typeface = typeface
                                }
                                binding.editor.apply {
                                  typefaceText = typeface
                                  setLineSpacing(2f, 1.1f)
                        cursorAnimator = ScaleCursorAnimator(this)
                                nonPrintablePaintingFlags =
    CodeEditor.FLAG_DRAW_WHITESPACE_LEADING or CodeEditor.FLAG_DRAW_LINE_SEPARATOR or CodeEdi
                                // Update display dynamically
           subscribeEvent<SelectionChangeEvent> { _, _ -> updatePositionText() }
                        subscribeEvent<ContentChangeEvent> { _, _ ->
                                    postDelayed(
                                    ::updateBtnState,
                                        50
                                    )
                                }
                        subscribeEvent<KeyBindingEvent> { event, _ ->
                        if (event.eventType != EditorKeyEvent.Type.DOWN) {
                                    return@subscribeEvent
                                        }
                                    }
                                }

                            if (savedInstanceState != null) {
                    binding.editor.setText(savedInstanceState.getString(TEXT_KEY))
                                } else {
                            binding.editor.setText(currentFile.readText())
                                }
                                updatePositionText()
                                  updateBtnState()
                        CoroutineScope(Dispatchers.Default).launch {
                setTheme(dataStore.mThemeString.first() ?: EditorTheme.QuietLight)
                                }
                            binding.runCode.setOnClickListener {
                                    saveFile()
                                val mIntent = intent
                        mIntent.setClass(this, TermActivity::class.java)
                    mIntent.putExtra(Keys.KEY_FILE_PATH, currentFile.absolutePath)
                                startActivity(mIntent)
                                    }
                                }

                        private fun setCurrentFile() {
                     if (intent.getStringExtra(Keys.KEY_FILE_PATH) != null) {
                                    try {
                      currentFile = File(intent.getStringExtra(Keys.KEY_FILE_PATH)!!)
                              binding.editor.setText(currentFile.readText())
                                if (!currentFiles.contains(currentFile)) {
                                 currentFiles = currentFiles.plus(currentFile)
                                        }
                                    }
                                catch (e:Exception){
                                   e.printStackTrace()
                      Toast.makeText(this,"File not found",Toast.LENGTH_LONG).show()
                                     finish()
                                      }
                                   } else {
                                val uri = intent.data
                                  if (uri != null) {
                                    try {
                 val filePath: String = UriUtils.uri2FileNoCacheCopy(uri).absolutePath
                                if (filePath.endsWith(".py")) {
                                    currentFile = File(filePath)
                            binding.editor.setText(currentFile.readText())
                             currentFiles = currentFiles.plus(currentFile)
                                        }
                                  } catch (e: Exception) {
        if (e is ErrnoException || e is FileNotFoundException || e is SecurityException) {
                                    e.printStackTrace()
```

```
                        Toast.makeText(this, "Permission denied", Toast.LENGTH_LONG).show()
                                                finish()
                                            } else {
                                                try {
                                val filePath: String = UriUtils.uri2File(uri).absolutePath
                                Toast.makeText(this,"This file is read only", Toast.LENGTH_LONG).show()
                                                currentFile = File(filePath)
                                            binding.editor.setText(currentFile.readText())
                                            currentFiles = currentFiles.plus(currentFile)

                                                } catch (e2 : Exception) {
                                                    e2.printStackTrace()
                                                        finish()
                                                      }
                                                    }
                                                  }
                                                }
                                              }

                                        private fun updateBtnState() {
                                    undo?.isEnabled = binding.editor.canUndo()
                                    redo?.isEnabled = binding.editor.canRedo()
                                                }

                                      private fun updatePositionText() {
                                        val cursor = binding.editor.cursor
                                                var text = ""
                                             text += if (cursor.isSelected) {
                                    "(" + (cursor.right - cursor.left) + " chars)"
                                                } else {
                                                    val content = binding.editor.text
                            if (content.getColumnCount(cursor.leftLine) == cursor.leftColumn) {
                                  "(<" + content.getLine(cursor.leftLine).lineSeparator.let {
                                                if (it == LineSeparator.NONE) {
                                                        "EOF"
                                                    } else {
                                                       it.name
                                                        }
                                                    } + ">)"
                                                } else {
                                            val char = binding.editor.text.charAt(
                                                        cursor.leftLine,
                                                        cursor.leftColumn
                                                            )
                                               if (char.isLowSurrogate() && cursor.leftColumn > 0) {
                                                            "(" + String(
                                                              charArrayOf(
                                                        binding.editor.text.charAt(
                                                               cursor.leftLine,
                                                            cursor.leftColumn - 1
                                                                 ), char
                                                                  )
                                                             ) + ")"
                        } else if (char.isHighSurrogate() && cursor.leftColumn + 1 < binding.editor.text.getColum
                                                           cursor.leftLine
                                                              )
                                                             ) {
                                                           "(" + String(
                                                             charArrayOf(
                                                        char, binding.editor.text.charAt(
                                                              cursor.leftLine,
                                                             cursor.leftColumn + 1
                                                                    )
                                                                 )
                                                             ) + ")"
                                                           } else {
                                                        "(" + escapeIfNecessary(
                                                         binding.editor.text.charAt(
                                                                cursor.leftLine,
                                                                cursor.leftColumn
                                                                      )
                                                             ) + ")"
                                                              }
                                                            }
                                                          }
                                                        }

                                    private fun escapeIfNecessary(c: Char): String {
                                                    return when (c) {
                                                        '\n' -> "\\n"
                                                        '\t' -> "\\t"
                                                        '\r' -> "\\r"
                                                        ' ' -> "<ws>"
                                                    else -> c.toString()
                                                            }
                                                        }

                                override fun onCreateOptionsMenu(menu: Menu): Boolean {
                                        menuInflater.inflate(R.menu.menu_main, menu)
                                            undo = menu.findItem(R.id.text_undo)
                                            redo = menu.findItem(R.id.text_redo)
                                           return super.onCreateOptionsMenu(menu)
                                                    }

                                  override fun onSaveInstanceState(outState: Bundle) {
```

```
                                super.onSaveInstanceState(outState)
                            outState.putString(TEXT_KEY, binding.editor.text.toString())
                                                }

                                        override fun onDestroy() {
                                               super.onDestroy()
                                           binding.editor.release()
                                                }

                        override fun onOptionsItemSelected(item: MenuItem): Boolean {
                                             val id = item.itemId
                                          val editor = binding.editor
                                                  when (id) {
                                          R.id.text_undo -> editor.undo()
                                       R.id.current_files -> showCurrentListDialog()
                                        R.id.text_file_add -> showFileListDialog()
                                          R.id.text_redo -> editor.redo()
                                            R.id.text_save -> saveFile()
                                                R.id.text_theme -> {
                             showThemeDialog(arrayOf("Quiet Light", "Darcula", "Abyss Color"))
                                                    }
                                                }
                                       return super.onOptionsItemSelected(item)
                                                }

                                             private fun saveFile() {
                                          val fos = FileOutputStream(currentFile)
                                           CoroutineScope(Dispatchers.IO).run {
                                    fos.write(binding.editor.text.toString().toByteArray())
                                                  fos.close()
                                                    }
                                Toast.makeText(this, "Saved", Toast.LENGTH_SHORT).show()
                                                }

                                    private fun showCurrentListDialog() {
                                   val builder = MaterialAlertDialogBuilder(this)
                                         builder.setTitle("Recent Files")
                                          if (currentFiles.isEmpty()) {
                            Toast.makeText(this, "No files found", Toast.LENGTH_SHORT).show()
                                                  return
                                                  }
builder.setSingleChoiceItems(currentFiles.map { it.name }.toTypedArray(),currentFiles.indexOf(cur
                                       currentFile = currentFiles[which]
                                    binding.editor.setText(currentFile.readText())
                                      if (!currentFiles.contains(currentFile)) {
                                       currentFiles = currentFiles.plus(currentFile)
                                                    }
                                                dialog?.dismiss()
                                                    }
                                           val dialog = builder.create()
                                                   dialog.show()
                                                    }
 private fun showFileListDialog(files: List<File> = PythonFileManager.pythonFiles.value) {
                                val builder = MaterialAlertDialogBuilder(this)
                                      builder.setTitle("Choose File")
                                          if (files.isEmpty()) {
                            Toast.makeText(this, "No files found", Toast.LENGTH_SHORT).show()
                                                  return
                                                  }
  builder.setItems(files.map { it.name }.toTypedArray()) { _: DialogInterface?, which: Int ->
                                       currentFile = files[which]
                                    binding.editor.setText(currentFile.readText())
                                      if (!currentFiles.contains(currentFile)) {
                                       currentFiles = currentFiles.plus(currentFile)
                                                    }
      Toast.makeText(this, "File changed to ${files[which].name}", Toast.LENGTH_SHORT).show()
                                                    }
                                           val dialog = builder.create()
                                                   dialog.show()
                                                    }
                         private fun showThemeDialog(themes: Array<String>) {
                               val builder = MaterialAlertDialogBuilder(this)
                                      builder.setTitle("Choose Theme")
                        builder.setItems(themes) { _: DialogInterface?, which: Int ->
                                         when (which) {
                                                0 -> {
                                        setTheme(EditorTheme.QuietLight)
                                     CoroutineScope(Dispatchers.IO).launch {
                                    dataStore.updateTheme(EditorTheme.QuietLight)
                                                    }
                                                  }

                                                1 -> {
                                           setTheme(EditorTheme.DarculaTheme)
                                     CoroutineScope(Dispatchers.IO).launch {
                                   dataStore.updateTheme(EditorTheme.DarculaTheme)
                                                    }
                                                  }

                                                2 -> {
                                        setTheme(EditorTheme.AbyssColor)
                                     CoroutineScope(Dispatchers.IO).launch {
                                    dataStore.updateTheme(EditorTheme.AbyssColor)
                                                    }
                                                  }
                                                }
```

```
                                        }
                                val dialog = builder.create()
                                        dialog.show()
                                            }

                            private fun setTheme(mTheme: String) {
                                    val editor = binding.editor
                                           when (mTheme) {
                                    EditorTheme.QuietLight -> try {
                                 val themeSource = IThemeSource.fromInputStream(
                                     assets.open("textmate/QuietLight.tmTheme"),
                                               "QuietLight.tmTheme",
                                                       null
                                                      )
                                val colorScheme = TextMateColorScheme.create(themeSource)
                                             editor.colorScheme = colorScheme
                                         val language = TextMateLanguage.create(
                                               IGrammarSource.fromInputStream(
                                   assets.open("textmate/python/syntax/python.tmLanguage.json"),
                                               "Python.tmLanguage.json",
                                                        null
                                                       ),
                        InputStreamReader(assets.open("textmate/python/language-configuration.json")),
                                    (editor.colorScheme as TextMateColorScheme).themeSource
                                                      )
                                            editor.setEditorLanguage(language)
                                               } catch (e: Exception) {
                                                    e.printStackTrace()
                                                           }

                                           EditorTheme.DarculaTheme -> try {
                                 val themeSource = IThemeSource.fromInputStream(
                                               assets.open("textmate/darcula.json"),
                                                     "darcula.json",
                                                           null
                                                          )

                                val colorScheme = TextMateColorScheme.create(themeSource)
                                             editor.colorScheme = colorScheme
                                         val language = TextMateLanguage.create(
                                               IGrammarSource.fromInputStream(
                                   assets.open("textmate/python/syntax/python.tmLanguage.json"),
                                               "Python.tmLanguage.json",
                                                        null
                                                       ),
                        InputStreamReader(assets.open("textmate/python/language-configuration.json")),
                                    (editor.colorScheme as TextMateColorScheme).themeSource
                                                      )
                                            editor.setEditorLanguage(language)
                                               } catch (e: Exception) {
                                                    e.printStackTrace()
                                                           }

                                           EditorTheme.AbyssColor -> try {
                                 val themeSource = IThemeSource.fromInputStream(
                                               assets.open("textmate/abyss-color-theme.json"),
                                                     "abyss-color-theme.json",
                                                           null
                                                          )
                                val colorScheme = TextMateColorScheme.create(themeSource)
                                             editor.colorScheme = colorScheme
                                         val language = TextMateLanguage.create(
                                               IGrammarSource.fromInputStream(
                                   assets.open("textmate/python/syntax/python.tmLanguage.json"),
                                               "Python.tmLanguage.json",
                                                        null
                                                       ),
                        InputStreamReader(assets.open("textmate/python/language-configuration.json")),
                                    (editor.colorScheme as TextMateColorScheme).themeSource
                                                      )
                                            editor.setEditorLanguage(language)
                                               } catch (e: Exception) {
                                                    e.printStackTrace()
                                                           }
                                                       }
                                                   }

                                            companion object {
                                     private const val TAG = "EditorActivity"
                                     private const val TEXT_KEY = "SavedText"
                                                   }

                                                }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/activities/TermActivity.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/activities/TermActivity.kt

File Type: Kotlin

Lines of Code: 326

```kotlin
package com.ankit.pythonpocketide.activities

import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.view.KeyEvent
import android.view.MotionEvent
import androidx.activity.ComponentActivity
import androidx.activity.addCallback
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.viewinterop.AndroidView
import androidx.core.content.res.ResourcesCompat
import com.blankj.utilcode.util.ClipboardUtils
import com.blankj.utilcode.util.KeyboardUtils
import com.termux.terminal.BuildConfig
import com.termux.terminal.TerminalEmulator
import com.termux.terminal.TerminalSession
import com.termux.terminal.TerminalSessionClient
import com.termux.view.TerminalRenderer
import com.termux.view.TerminalView
import com.termux.view.TerminalViewClient
import com.ankit.pythonpocketide.R
import com.ankit.pythonpocketide.ui.theme.PythonPocketIDETheme
import com.ankit.pythonpocketide.utils.Commands
import com.ankit.pythonpocketide.utils.Keys
import java.io.File
import java.lang.ref.WeakReference


class TermActivity : ComponentActivity(),TerminalViewClient {
    private lateinit var mTermView : TerminalView
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent{
            PythonPocketIDETheme {
                TermScreen()
            }
        }
        onBackPressedDispatcher.addCallback(this,true) {
            mTermView.mTermSession.finishIfRunning()
            finish()
        }
    }

    @Composable
    fun TermScreen()
    {
        AndroidView(
            modifier = Modifier.fillMaxSize(),
            factory = { context ->
                mTermView = TerminalView(context , null)
                Log.d(TAG,"Terminal has been created")
                mTermView.attachSession(getTerminalSession())
                mTermView.setTerminalViewClient(this)
mTermView.mRenderer= TerminalRenderer(32,ResourcesCompat.getFont(this, R.font.jetbrainsmo
                val fileName = intent.getStringExtra(Keys.KEY_FILE_PATH)
                val projectPath = intent.getStringExtra(Keys.KEY_PROJECT_PATH)
                val isProjectRun = intent.getBooleanExtra(Keys.KEY_PROJECT_RUN, false)
                val isPipInstall = intent.getBooleanExtra(Keys.KEY_PIP_INSTALL, false)
                val pipPackage = intent.getStringExtra(Keys.KEY_PIP_PACKAGE)

                if (fileName != null) {
                    // Enhanced file execution with better pip support
                    val command = if (isPipInstall && pipPackage != null) {
                        // Improved pip installation command
                        Commands.getEnhancedPipInstallCommand(this, pipPackage)
                    } else {
                        // Enhanced Python execution command
                        Commands.getEnhancedInterpreterCommand(this, fileName)
                    }
                    Log.d(TAG, "Running enhanced command: $command")
                    Handler(Looper.getMainLooper()).post {
                        mTermView.mTermSession.write("$command\r")
                    }
                } else if (intent.getBooleanExtra(Keys.IS_SHELL_MODE_KEY, false)) {
                    // Enhanced Python interactive shell
                    Handler(Looper.getMainLooper()).post {
                        mTermView.mTermSession.write("${Commands.getEnhancedPythonShellCommand(this)}\r")
                    }
                } else {
```

```
                    // Enhanced basic terminal with better environment
                        Handler(Looper.getMainLooper()).post {
                mTermView.mTermSession.write("${Commands.getEnhancedBasicCommand(this)}\r")
                                            }
                                        }
                mTermView.setBackgroundColor(this.getColor(R.color.terminal_colour))
                                    mTermView
                                      }
                                    )
                                  }

   private fun getTerminalSession(): TerminalSession {
                val cwd = filesDir.absolutePath
                      var shell = "/bin/sh"
              if (File("/bin/sh").exists().not())
                                    {
                         shell="/system/bin/sh"
                                    }
                     return TerminalSession(
                               shell,
                       cwd, arrayOf<String>(),
                             arrayOf(),
          TerminalEmulator.DEFAULT_TERMINAL_TRANSCRIPT_ROWS,
                         getTermSessionClient()
                                    )
                                  }
    override fun logError(tag: String?, message: String?) {
            if (BuildConfig.DEBUG && message != null) {
                          Log.e(tag, message)
                                    }
                                  }

    override fun logWarn(tag: String?, message: String?) {
            if (BuildConfig.DEBUG && message != null) {
                          Log.w(tag, message)
                                    }
                                  }

    override fun logInfo(tag: String?, message: String?) {
            if (BuildConfig.DEBUG && message != null) {
                          Log.i(tag, message)
                                    }
                                  }

    override fun logDebug(tag: String?, message: String?) {
            if (BuildConfig.DEBUG && message != null) {
                          Log.d(tag, message)
                                    }
                                  }

   override fun logVerbose(tag: String?, message: String?) {
            if (BuildConfig.DEBUG && message != null) {
                          Log.v(tag, message)
                                    }
                                  }

            override fun logStackTraceWithMessage(
                          tag: String?,
                        message: String?,
                          e: Exception?
                                ) {
                       if (BuildConfig.DEBUG) {
        Log.e(tag, message + "\n" + Log.getStackTraceString(e))
                                    }
                                  }

   override fun logStackTrace(tag: String?, e: Exception?) {
                       if (BuildConfig.DEBUG) {
                 Log.e(tag, Log.getStackTraceString(e))
                                    }
                                  }
          override fun onScale(scale: Float): Float {
                               return scale
                                  }

        override fun onSingleTapUp(e: MotionEvent?) {
             if (mTermView.mTermSession.isRunning) {
                       mTermView.requestFocus()
                  KeyboardUtils.showSoftInput(mTermView)
                                    }
                                  }

override fun shouldBackButtonBeMappedToEscape(): Boolean { return false }

    override fun shouldEnforceCharBasedInput(): Boolean {
                              return true
                                  }

    override fun shouldUseCtrlSpaceWorkaround(): Boolean {
                             return false
                                  }

       override fun isTerminalViewSelected(): Boolean {
                              return true
                                  }
```

```
                override fun copyModeChanged(copyMode: Boolean) { }

    override fun onKeyDown(keyCode: Int, e: KeyEvent?, session: TerminalSession?): Boolean {
                                        return false
                                            }

            override fun onKeyUp(keyCode: Int, e: KeyEvent?): Boolean {
                        if (keyCode == KeyEvent.KEYCODE_BACK) {
                          if (mTermView.mTermSession.isRunning) {
                            mTermView.mTermSession.finishIfRunning()
                                           finish()
                                            }
                                       return true
                                          }
                                        return false
                                            }

             override fun onLongPress(event: MotionEvent?): Boolean {
                                        return false
                                            }

                     override fun readControlKey(): Boolean {
                                        return false
                                            }

                       override fun readAltKey(): Boolean {
                                        return false
                                            }

                      override fun readShiftKey(): Boolean {
                                        return false
                                            }

                       override fun readFnKey(): Boolean {
                                        return false
                                            }

                           override fun onCodePoint(
                                  codePoint: Int,
                                ctrlDown: Boolean,
                              session: TerminalSession?
                                   ): Boolean {
                                     return false
                                       }
                private fun getTermSessionClient() : TerminalSessionClient
                                       {
                       val weakActivityReference = WeakReference(this)
                           return object : TerminalSessionClient {
                  override fun onTextChanged(changedSession: TerminalSession) {
                                    runOnUiThread {
                        weakActivityReference.get()?.mTermView?.onScreenUpdated()
                                           }
                                       }

                override fun onTitleChanged(updatedSession: TerminalSession) {  }

                override fun onSessionFinished(finishedSession: TerminalSession) {
                                    runOnUiThread{
                               weakActivityReference.get()?.mTermView?.let {
                                       KeyboardUtils.hideSoftInput(it)
                                      it.mTermSession?.finishIfRunning()
                                                }
                                           finish()
                                             }
                                       }

        override fun onCopyTextToClipboard(session: TerminalSession, text: String?) {
                                  ClipboardUtils.copyText(text)
                                       }

                override fun onPasteTextFromClipboard(session: TerminalSession?) {
                                    runOnUiThread{
                             val clip = ClipboardUtils.getText().toString()
                                   if (clip.trim { it <= ' ' }
                   .isNotEmpty() && weakActivityReference.get()?.mTermView?.mEmulator != null) {
                        weakActivityReference.get()?.mTermView?.mEmulator?.paste(clip)
                                           }
                                             }
                                       }

                        override fun onBell(session: TerminalSession) { }

                override fun onColorsChanged(changedSession: TerminalSession) { }

                  override fun onTerminalCursorStateChange(state: Boolean) { }

                           override fun getTerminalCursorStyle(): Int {
                        return TerminalEmulator.TERMINAL_CURSOR_STYLE_UNDERLINE
                                       }

                     override fun logError(tag: String?, message: String?) {
                             if (BuildConfig.DEBUG && message != null) {
                                       Log.e(tag, message)
                                         }
                                       }
```

```
                                override fun logWarn(tag: String?, message: String?) {
                                        if (BuildConfig.DEBUG && message != null) {
                                                        Log.w(tag, message)
                                                            }
                                                        }

                                override fun logInfo(tag: String?, message: String?) {
                                        if (BuildConfig.DEBUG && message != null) {
                                                        Log.i(tag, message)
                                                            }
                                                        }

                                override fun logDebug(tag: String?, message: String?) {
                                        if (BuildConfig.DEBUG && message != null) {
                                                        Log.d(tag, message)
                                                            }
                                                        }

                               override fun logVerbose(tag: String?, message: String?) {
                                        if (BuildConfig.DEBUG && message != null) {
                                                        Log.v(tag, message)
                                                            }
                                                        }

                                         override fun logStackTraceWithMessage(
                                                        tag: String?,
                                                      message: String?,
                                                        e: Exception?
                                                           ) {
                                                    if (BuildConfig.DEBUG) {
                                    Log.e(tag, message + "\n" + Log.getStackTraceString(e))
                                                            }
                                                        }

                               override fun logStackTrace(tag: String?, e: Exception?) {
                                                    if (BuildConfig.DEBUG) {
                                             Log.e(tag, Log.getStackTraceString(e))
                                                            }
                                                        }

                                                      }
                                                     }
                                        override fun onEmulatorSet() { }
                                                companion object {
                                       private const val TAG = "TermActivity"
                                                      }
                                                    }
                                                   /*
H=$PATH:/data/app/~~Hfk1Sq2A4XNtLfCsC9OZSQ==/com.ankit.pythonpocketide-oxz5xUgjGGvDM-ewPK-G0A==/lib/arm64
                                                   */
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/activities/HomeActivity.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/activities/HomeActivity.kt

File Type: Kotlin

Lines of Code: 109

```
                    package com.ankit.pythonpocketide.activities
                           import android.content.Intent
                              import android.os.Bundle
                               import android.util.Log
                     import androidx.activity.ComponentActivity
                     import androidx.activity.compose.setContent
               import androidx.compose.animation.ExperimentalAnimationApi
                  import androidx.compose.foundation.layout.Arrangement
                     import androidx.compose.foundation.layout.Column
                  import androidx.compose.foundation.layout.fillMaxSize
               import androidx.compose.material3.CircularProgressIndicator
                        import androidx.compose.material3.Text
                     import androidx.compose.runtime.mutableStateOf
                          import androidx.compose.ui.Alignment
                           import androidx.compose.ui.Modifier
import com.google.accompanist.navigation.material.ExperimentalMaterialNavigationApi
                           import com.hzy.libp7zip.P7ZipApi
            import com.ramcosta.composedestinations.DestinationsNavHost
import com.ramcosta.composedestinations.animations.defaults.RootNavGraphDefaultAnimations
    import com.ramcosta.composedestinations.animations.rememberAnimatedNavHostEngine
              import com.ramcosta.composedestinations.navigation.dependency
              import com.ankit.pythonpocketide.dataStore.SettingsDataStore
             import com.ankit.pythonpocketide.ui.theme.PythonPocketIDETheme
                       import com.ankit.pythonpocketide.utils.Keys
                 import com.ankit.pythonpocketide.utils.PythonFileManager
                        import kotlinx.coroutines.CoroutineScope
                          import kotlinx.coroutines.Dispatchers
                          import kotlinx.coroutines.flow.first
                            import kotlinx.coroutines.launch
                                   import java.io.File
                               import java.nio.file.Files
                        import java.nio.file.StandardCopyOption

    @OptIn(ExperimentalAnimationApi::class, ExperimentalMaterialNavigationApi::class)
                        class HomeActivity: ComponentActivity() {
                      private lateinit var dataStore:SettingsDataStore
                     private val isFileExtracting= mutableStateOf(false)
      @OptIn(ExperimentalAnimationApi::class, ExperimentalMaterialNavigationApi::class)
                     override fun onCreate(savedInstanceState: Bundle?) {
                                super.onCreate(savedInstanceState)
                           dataStore=SettingsDataStore(applicationContext)
                        // create a directory for python files if not exists
                    val pythonFilesDir = File(filesDir.absolutePath+"/pythonFiles")
                                if (!pythonFilesDir.exists()) {
                                       pythonFilesDir.mkdir()
                                                }
                          PythonFileManager.filesDir=pythonFilesDir.absolutePath
                                        PythonFileManager.init()
                                              setContent()
                                                      {
                                             PythonPocketIDETheme()
                                                      {
                                           if (isFileExtracting.value){
          Column(modifier = Modifier.fillMaxSize(), verticalArrangement = Arrangement.Center, h
                                                CircularProgressIndicator()
                                             Text(text = "Extracting files...")
                                                        }
                                                      }
                                                     else{
                                    val navHostEngine = rememberAnimatedNavHostEngine(
                                        navHostContentAlignment = Alignment.TopCenter,
                        rootDefaultAnimations = RootNavGraphDefaultAnimations.ACCOMPANIST_FADING)

                                                 DestinationsNavHost(
                                    navGraph = com.ankit.pythonpocketide.ui.screens.NavGraphs.root,
                                    dependenciesContainerBuilder = { dependency(this@HomeActivity) },
                                                    engine = navHostEngine
                                                      )
                                                    }
                                                   }
                                                  }
                                               extractFiles()
                                                   }
                                    private fun extractFiles() {
                                      isFileExtracting.value = true
                                   CoroutineScope(Dispatchers.IO).launch {
                                if (dataStore.areFilesExtracted.first() == true) {
                                            isFileExtracting.value = false
                                                     } else {
                                               dataStore.updateFileStatus(false)
                                             CoroutineScope(Dispatchers.IO).launch {
                                             val temp7zStream = assets.open("python.7z")
                                  val file = File("${filesDir.absolutePath}/python.7z")
```

```
                                        file.createNewFile()
                Files.copy(temp7zStream,file.toPath(),StandardCopyOption.REPLACE_EXISTING)
        val exitCode = P7ZipApi.executeCommand("7z x ${file.absolutePath} -o${filesDir.absolu
                            Log.d(TAG, "extractFiles: $exitCode")
                                        file.delete()
                                    temp7zStream.close()
                                CoroutineScope(Dispatchers.IO).launch {
                                     dataStore.updateFileStatus(true)
                                                }
                                    isFileExtracting.value = false
                                            }
                                        }
                                    }
                                }
                            fun startEditorActivity(file: File)
                                {
                     val mIntent = Intent(this, EditorActivity::class.java)
                mIntent.putExtra(Keys.KEY_FILE_PATH,file.absoluteFile.toString())
                                startActivity(mIntent)
                                }

                                companion object {
                            const val TAG = "WelcomeActivity"
                                }
                            }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/screens/SampleScreen.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/screens/SampleScreen.kt

File Type: Kotlin

Lines of Code: 88

```
package com.ankit.pythonpocketide.ui.screens

import android.content.Context
import androidx.compose.foundation.background
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Divider
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBar
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.font.Font
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.unit.dp
import com.ramcosta.composedestinations.annotation.Destination
import com.ankit.pythonpocketide.R
import com.ankit.pythonpocketide.activities.HomeActivity
import com.ankit.pythonpocketide.ui.layoutComponents.ListComponent
import java.io.File
import java.nio.file.Files
import java.nio.file.StandardCopyOption

@OptIn(ExperimentalMaterial3Api::class)
@Destination
@Composable
fun SampleScreen(welcomeActivity: HomeActivity) {
val sampleFiles by remember { mutableStateOf(welcomeActivity.assets.list("Samples")) }
Scaffold(
topBar =
{
Surface(shadowElevation = 10.dp){
TopAppBar(
title = {
Text(
text = "Samples",
fontFamily = FontFamily(Font(resId = R.font.roboto_condensed_bold))
)
}
)
}
}
) {
LazyColumn(
Modifier.padding(it)
)
{
items(sampleFiles!!.size)
{ index ->
val file = getAssetFile(welcomeActivity, sampleFiles!![index])
ListComponent(
file = file,
modifier = Modifier
.padding(5.dp)
.fillMaxWidth()
.background(
color = MaterialTheme.colorScheme.primary.copy(alpha = 0.15f),
shape = RoundedCornerShape(8.dp)
)
.clickable { welcomeActivity.startEditorActivity(file) }
)
Divider(Modifier.fillMaxWidth(1f))
}
}
}
}

fun getAssetFile(context: Context, assetName: String): File {
val file = File(context.cacheDir, assetName)
//draw image if created successfully
file.createNewFile()
Files.copy(
context.assets.open("Samples/$assetName"),
file.toPath(),
```

```
                StandardCopyOption.REPLACE_EXISTING
                )
            return file
        }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/screens/HomeScreen.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/screens/HomeScreen.kt

File Type: Kotlin

Lines of Code: 261

```kotlin
package com.ankit.pythonpocketide.ui.screens

import android.annotation.SuppressLint
import android.content.Intent
import android.content.pm.PackageManager
import android.os.Build
import android.os.Environment
import android.widget.Toast
import androidx.compose.foundation.background
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.Font
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.unit.dp
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import androidx.lifecycle.ViewModelProvider
import com.ramcosta.composedestinations.annotation.Destination
import com.ramcosta.composedestinations.annotation.RootNavGraph
import com.ramcosta.composedestinations.navigation.DestinationsNavigator
import com.ankit.pythonpocketide.R
import com.ankit.pythonpocketide.activities.HomeActivity
import com.ankit.pythonpocketide.activities.TermActivity
import com.ankit.pythonpocketide.ui.layoutComponents.FullScreenMessage
import com.ankit.pythonpocketide.ui.layoutComponents.ListComponent
import com.ankit.pythonpocketide.ui.layoutComponents.MenuItem
import com.ankit.pythonpocketide.ui.screens.destinations.AboutScreenDestination
import com.ankit.pythonpocketide.ui.screens.destinations.FilePickerScreenDestination
import com.ankit.pythonpocketide.ui.screens.destinations.LibraryDownloaderScreenDestination
import com.ankit.pythonpocketide.ui.screens.destinations.SampleScreenDestination
import com.ankit.pythonpocketide.utils.Keys
import com.ankit.pythonpocketide.utils.PermissionManageExternal
import com.ankit.pythonpocketide.utils.PythonFileManager
import com.ankit.pythonpocketide.viewModels.HomeScreenViewModel
import kotlinx.coroutines.launch

@OptIn(ExperimentalMaterial3Api::class)
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter", "CoroutineCreationDuringComposition")
@RootNavGraph(start = true)
@Destination
@Composable
fun HomeScreen(welcomeActivity: HomeActivity, navigator: DestinationsNavigator) {
    val mViewModel = ViewModelProvider(welcomeActivity)[HomeScreenViewModel::class.java]
    val scope = rememberCoroutineScope()
    val pythonFilesList by  PythonFileManager.pythonFiles
    val items = listOf(
        MenuItem("Terminal", "Terminal", R.drawable.terminal_icon) {
            welcomeActivity.startActivity(Intent(welcomeActivity, TermActivity::class.java))
            scope.launch { mViewModel.mDrawerState.value.close() }
        },
        MenuItem("Libraries", "Libraries", R.drawable.library_icon) {
            navigator.navigate(LibraryDownloaderScreenDestination)
            scope.launch { mViewModel.mDrawerState.value.close() }
        },
        MenuItem("Samples", "Samples", R.drawable.sample_icon) {
            navigator.navigate(SampleScreenDestination)
            scope.launch { mViewModel.mDrawerState.value.close() }
        },
        MenuItem(
            "PythonShell",
            "Interactive Mode",
            R.drawable.interactive_mode_icon
        ) {
            val intent = Intent(welcomeActivity, TermActivity::class.java)
            intent.putExtra(Keys.IS_SHELL_MODE_KEY, true)
            welcomeActivity.startActivity(intent)
            scope.launch { mViewModel.mDrawerState.value.close() }
        },
        MenuItem("New File", "Create new file", R.drawable.create_file_icon) {
            scope.launch {
                mViewModel.showDialog()
            }
        },
        MenuItem("Open File", "Open file", R.drawable.file_open_icon) {
            if (Build.VERSION.SDK_INT < Build.VERSION_CODES.R){
```

```
                                        ActivityCompat.requestPermissions(
                                                welcomeActivity,
                                    arrayOf(android.Manifest.permission.WRITE_EXTERNAL_STORAGE),
                                                    1
                                                )
                                        if (ContextCompat.checkSelfPermission(
                                                    welcomeActivity,
                                            android.Manifest.permission.WRITE_EXTERNAL_STORAGE
                                                ) == PackageManager.PERMISSION_GRANTED
                                                ) {
                                            navigator.navigate(FilePickerScreenDestination)
                                                }
                                            }
                                           else {
                                        if (Environment.isExternalStorageManager()){
                                            navigator.navigate(FilePickerScreenDestination)
                                                }
                                                else {
                    Toast.makeText(welcomeActivity,"Please grant storage permission",Toast.LENGTH_SHORT).s
                                            PermissionManageExternal.request(welcomeActivity)
                                                }
                                            }
                                           },
                    MenuItem("Info", "About", R.drawable.about_icon) {
                                  navigator.navigate(AboutScreenDestination)
                                scope.launch { mViewModel.mDrawerState.value.close() }
                                            }
                                        )
                                    ModalNavigationDrawer(
                               drawerState = mViewModel.mDrawerState.value,
                                            drawerContent = {
                                              ModalDrawerSheet(
                                               modifier = Modifier
                                                      .sizeIn(
                                                   minWidth = 240.dp,
                                                    maxWidth = 290.dp
                                                        )
                                                   .fillMaxHeight()
                                                    ) {
                                            Spacer(Modifier.height(12.dp))
                                                items.forEach { item ->
                                                   NavigationDrawerItem(
                                                           icon = {
                                                               Icon(
                                                painter = painterResource(id = item.resID),
                                                           contentDescription = null
                                                                )
                                                            },
                                                label = { Text(item.title) },
                                                      selected = false,
                                                 onClick = item.clickable,
                                        modifier = Modifier.padding(NavigationDrawerItemDefaults.ItemPadding)
                                                )
                                            }
                                           }
                                        ) {
                                         Scaffold(
                                           topBar =
                                               {
                                            TopAppBar(title = {
                                                     Text(
                                                text = "Python Pocket IDE",
                                     fontFamily = FontFamily(Font(resId = R.font.roboto_condensed_bold))
                                                )
                                                   },
                                              modifier = Modifier
                                                .padding(10.dp, 0.dp, 10.dp, 2.dp)
                                                ,
                                                 navigationIcon = {
                                                        Icon(
                                                    Icons.Filled.Menu,
                                                  contentDescription = "Menu",
                                    modifier = Modifier.clickable { scope.launch { mViewModel.mDrawerState.value.(
                                                   },
                                                actions = {
                                            Icon(Icons.Default.Info, contentDescription = "Info",
                                                    Modifier
                                                    .size(26.dp)
                                             .clickable { navigator.navigate(AboutScreenDestination) })
                                                   })
                                                },
                                           floatingActionButton =
                                                {
                                              FloatingActionButton(
                                                   onClick = {
                                                    scope.launch {
                                                mViewModel.showDialog()
                                                    }
                                                },
                                  content = { Icon(Icons.Filled.Add, contentDescription = "Plus button") },
                                                )
                                            }
                                           ) {
                                    if (pythonFilesList.isNotEmpty()) {
                                                LazyColumn(
```

```
                                    Modifier.padding(it)
                                            )
                                            {
                                items(pythonFilesList.size)
                                        { index ->
                                        ListComponent(
                                   file = pythonFilesList[index],
                                         modifier = Modifier
                                              .padding(5.dp)
                                             .fillMaxWidth()
                                              .background(
                        color = MaterialTheme.colorScheme.primary.copy(alpha = 0.15f),
                                shape = RoundedCornerShape(8.dp)
                                                    )
.clickable { welcomeActivity.startEditorActivity(pythonFilesList[index]) }
                                              )
                            Divider(Modifier.fillMaxWidth(1f))
                                            }

                                        }
                                    } else {
                                FullScreenMessage(
                     icon = painterResource(id = R.drawable.file_icon),
                                title = "No Python Files",
                     message = "Create by clicking on the \"+\" button"
                                            )
                                    }
                       if (mViewModel.mDialogState.value) {
                                    AlertDialog(
                                  onDismissRequest =
                                            {
                                mViewModel.dismissDialog()
                                            },
                                   confirmButton =
                                            {
                                        TextButton(
                                          onClick = {
                                           scope.launch {
                           mViewModel.saveFile(mViewModel.mFileName.value)
                                    mViewModel.dismissDialog()
                                mViewModel.changeFileName("")
                                             }}
                                            ) {
                                         Text("Save")
                                            }
                                            },
                                  dismissButton =
                                            {
                                        TextButton(
                            onClick = { mViewModel.dismissDialog()}
                                            ) {
                                         Text("Cancel")
                                             }
                                           },
                                       text = {
                                        TextField(
                              value = mViewModel.mFileName.value,
                                        onValueChange =
                                         { it:String->
                                           scope.launch {
                                mViewModel.changeFileName(it)
                                                }
                                             },
                                         maxLines = 1,
                                keyboardOptions = KeyboardOptions()
                                             )
                                           },
                                      title = {
                                 Text(text = "Enter file name")
                                           },
                                       icon = {
                                           Icon(
                    painter = painterResource(id = R.drawable.create_file_icon),
                                  contentDescription = null
                                            )
                                        }
                                      )
                                    }
                                  }
                                }
                              }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/screens/LibraryDownloaderScreen.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/screens/LibraryDownloaderScreen.kt

File Type: Kotlin

Lines of Code: 45

```
package com.ankit.pythonpocketide.ui.screens

import android.annotation.SuppressLint
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBar
import androidx.compose.runtime.Composable
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.Font
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.unit.dp
import com.ramcosta.composedestinations.annotation.Destination
import com.ankit.pythonpocketide.R
import com.ankit.pythonpocketide.ui.layoutComponents.FullScreenMessage

@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Destination
@Composable
fun LibraryDownloaderScreen()
{
Scaffold(
topBar =
{
Surface(shadowElevation = 10.dp){
TopAppBar(
title = {
Text(
text = "Libraries",
fontFamily = FontFamily(Font(resId = R.font.roboto_condensed_bold))
)
}
)
}
}
) {
FullScreenMessage(
icon = painterResource(id = R.drawable.coming_soon_icon),
title = "Coming Soon",
message = "Libraries like numpy, pandas, matplotlib, etc will be available soon."
)
}
}
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/screens/AboutScreen.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/screens/AboutScreen.kt

File Type: Kotlin

Lines of Code: 47

```
                        package com.ankit.pythonpocketide.ui.screens

                           import android.annotation.SuppressLint
                          import androidx.compose.foundation.layout.*
                              import androidx.compose.material3.*
                           import androidx.compose.runtime.Composable
                              import androidx.compose.ui.Modifier
                          import androidx.compose.ui.res.painterResource
                             import androidx.compose.ui.text.font.Font
                          import androidx.compose.ui.text.font.FontFamily
                                import androidx.compose.ui.unit.dp
                                import androidx.compose.ui.unit.sp
                  import com.ramcosta.composedestinations.annotation.Destination
                                import com.ankit.pythonpocketide.R

                     @SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
                             @OptIn(ExperimentalMaterial3Api::class)
                                          @Composable
                                          @Destination
                                       fun AboutScreen(){
                                           Scaffold(
                                            topBar =
                                              {
TopAppBar(title = { Text(text = "About", fontFamily =FontFamily(Font(R.font.roboto_condensed_bold))) }
                                modifier = Modifier.padding(10.dp,0.dp,10.dp,2.dp),
                                               actions = {
            Icon(painter = painterResource(id = R.drawable.app_icon), contentDescription = null)
                                                   })
                                             }
                                             ){
                                    Column(modifier = Modifier.padding(it))
                                                {
                                             Divider()
    Text(modifier = Modifier.padding(12.dp),text = "Python Pocket IDE (PPIDE)", fontSize = 16.sp, font
    Text(modifier = Modifier.padding(12.dp),text = "Developer = Ankit", fontSize = 14.sp, fontFamily =
    Text(modifier = Modifier.padding(12.dp),text = "Final Year Project - MCA", fontSize = 14.sp, fontF
    Text(modifier = Modifier.padding(12.dp),text = "Chandigarh University (Batch July 2023)", fontSize
                                Divider(modifier = Modifier.padding(12.dp))
    Text(modifier = Modifier.padding(20.dp),text = "Python Pocket IDE is an advanced mobile developme
                                Divider(modifier = Modifier.padding(12.dp))
    Text(text = "Based on KtxPy by PsiCodes (Pranjal)", fontSize = 13.sp, fontFamily = FontFamily(Font
    Text(text = "Special Thanks to Rosemoe, Termux team & hzy3774", fontSize = 13.sp, fontFamily = Fo
                                Divider(modifier = Modifier.padding(12.dp))
    Text(modifier = Modifier.padding(12.dp),text = "License: GPL v3", fontSize = 12.sp, fontFamily = F
    Text(modifier = Modifier.padding(12.dp),text = "Version: 2.0.0 (PPIDE Enhanced)", fontSize = 12.sp
                                              }
                                            }}
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/screens/FilePickerScreen.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/screens/FilePickerScreen.kt

File Type: Kotlin

Lines of Code: 144

```
package com.ankit.pythonpocketide.ui.screens

import android.content.Intent
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Info
import androidx.compose.material3.Divider
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Icon
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBar
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.Font
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.unit.dp
import com.ramcosta.composedestinations.annotation.Destination
import com.ankit.pythonpocketide.R
import com.ankit.pythonpocketide.activities.EditorActivity
import com.ankit.pythonpocketide.activities.HomeActivity
import com.ankit.pythonpocketide.ui.layoutComponents.ListComponent
import com.ankit.pythonpocketide.utils.Keys
import java.io.File


@OptIn(ExperimentalMaterial3Api::class)
@Destination
@Composable
fun FilePickerScreen() {
val context = LocalContext.current
val previousDirectory by remember{mutableStateOf(File("/storage/emulated/0/"))}
var directory by remember{mutableStateOf(File("/storage/emulated/0/"))}
val files = directory.listFiles { it ->
(it.isDirectory && it.name!="Android") || it.extension == "py" && !it.isHidden
} ?: emptyArray()
Scaffold(
topBar =
{
Surface(shadowElevation = 10.dp){
TopAppBar(
title =
{
Text(
text = "File Picker",
fontFamily = FontFamily(Font(resId = R.font.roboto_condensed_bold))
)
},
navigationIcon =
{
if (directory != previousDirectory) {
Icon(
painter = painterResource(id = R.drawable.back_icon),
contentDescription = "Previous directory",
modifier = Modifier.clickable
{
directory = previousDirectory
}
)
}
}
)
}
)
{
if (files.isNotEmpty())
{
LazyColumn(
```

```
                                    modifier = Modifier
                                          .padding(it)
                                           )
                                           {
                                    items(files.size) { index ->
                                     if (files[index].isDirectory) {
                                              ListComponent(
                                             file = files[index],
                                         icon = R.drawable.folder_icon,
                                              modifier = Modifier
                                                  .fillMaxWidth()
                                                  .padding(5.dp)
                                                   .clickable {
                                                directory = files[index]
                                                        }
                                                   )
                                                } else {
                                                ListComponent(
                                                file = files[index],
                                              icon = R.drawable.python_icon,
                                                    modifier = Modifier
                                                        .fillMaxWidth()
                                                        .padding(5.dp)
                                                          .clickable{
                                           val intent = Intent(context as HomeActivity, EditorActivity::class.ja
                                              intent.putExtra(Keys.KEY_FILE_PATH, files[index].absolutePath)
                                                        context.startActivity(intent)
                                                                }
                                                                )
                                                              }
                                                             Divider()
                                                               }
                                                              }
                                                             }
                                                            else
                                                              {
                                                             Column(
                                                          modifier = Modifier
                                                              .fillMaxSize(),
                                               verticalArrangement = Arrangement.Center,
                                            horizontalAlignment = Alignment.CenterHorizontally
                                                              )
                                                              {
                                                              Icon(
                                                    imageVector = Icons.Default.Info,
                                                  contentDescription = "No files found",
                                                           modifier = Modifier
                                                                .size(100.dp)
                                                               .padding(5.dp)
                                                                 )
                                                               Text(
                                                         text = "No files found",
                                          fontFamily = FontFamily(Font(resId = R.font.custom_sans)),
                                                  modifier = Modifier.padding(top = 2.dp),
                                                                  )
                                                                }
                                                               }
                                                              }

                                                             }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/layoutComponents/FullScreenMessage.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/layoutComponents/FullScreenMessage.kt

File Type: Kotlin

Lines of Code: 57

```
package com.ankit.pythonpocketide.ui.layoutComponents

    import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Arrangement
   import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
  import androidx.compose.foundation.layout.padding
    import androidx.compose.foundation.layout.size
        import androidx.compose.material3.Icon
   import androidx.compose.material3.MaterialTheme
        import androidx.compose.material3.Text
      import androidx.compose.runtime.Composable
         import androidx.compose.ui.Alignment
         import androidx.compose.ui.Modifier
 import androidx.compose.ui.graphics.painter.Painter
      import androidx.compose.ui.text.font.Font
   import androidx.compose.ui.text.font.FontFamily
         import androidx.compose.ui.unit.dp
         import androidx.compose.ui.unit.sp
         import com.ankit.pythonpocketide.R

                    @Composable
                fun FullScreenMessage(
                      icon: Painter,
                      title: String,
                     message: String
                           )
                           {
                         Column(
                     modifier = Modifier
                          .fillMaxSize()
        .background(MaterialTheme.colorScheme.background)
                             .padding(10.dp),
     horizontalAlignment = Alignment.CenterHorizontally,
           verticalArrangement = Arrangement.Center
                             )
                             {
                             Icon(
                           painter = icon,
                       contentDescription = null,
                   modifier = Modifier.size(64.dp),
            tint = MaterialTheme.colorScheme.primary
                             )
                            Text(
                           text = title,
          color = MaterialTheme.colorScheme.primary,
    fontFamily = FontFamily(Font(R.font.roboto_condensed_bold))
                             )
                            Text(
                           text = message,
          color = MaterialTheme.colorScheme.primary,
                        fontSize = 15.sp,
   fontFamily = FontFamily(Font(R.font.roboto_condensed_bold)),
   textAlign = androidx.compose.ui.text.style.TextAlign.Center
                             )
                           }
                          }
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/layoutComponents/ListComponent.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/layoutComponents/ListComponent.kt

File Type: Kotlin

Lines of Code: 67

```
package com.ankit.pythonpocketide.ui.layoutComponents

import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.material3.Icon
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.Font
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.ankit.pythonpocketide.R
import java.io.File
import java.text.SimpleDateFormat
import java.util.Date

@Composable
fun ListComponent(
modifier: Modifier = Modifier,
file : File ,
icon : Int = R.drawable.python_icon
){
Row(
modifier=modifier
)
{
Icon(painter = painterResource(icon),
contentDescription = null,
modifier = Modifier
.size(45.dp)
.padding(5.dp),
)
Column(){
Text(
"Name : ${file.name}",
fontFamily = FontFamily(Font(resId = R.font.custom_sans)),
modifier = Modifier.padding(top=2.dp),
maxLines = 1,
fontSize = 12.sp
)
Text(
text = "Last Modified : ${
SimpleDateFormat("dd-MM-yyyy", java.util.Locale.getDefault()).format(Date(file.lastMo
}",
maxLines = 1,
fontFamily = FontFamily(Font(resId = R.font.custom_sans)),
fontSize = 10.sp,
)
Text(
text = "Size : ${
if (file.length() > 1024 * 1024)
"${file.length() / (1024 * 1024)} MB"
else
"${file.length() / 1024} KB"
}",
maxLines = 1,
fontFamily = FontFamily(Font(resId = R.font.custom_sans)),
fontSize = 10.sp,
)
}
}

}
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/layoutComponents/MenuItem.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/layoutComponents/MenuItem.kt

File Type: Kotlin

Lines of Code: 6

```
package com.ankit.pythonpocketide.ui.layoutComponents

data class MenuItem(
val id:String, val title:String, val resID:Int, val clickable:()->Unit
)
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/theme/Theme.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/theme/Theme.kt

File Type: Kotlin

Lines of Code: 68

```
package com.ankit.pythonpocketide.ui.theme

import android.app.Activity
import android.os.Build
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.dynamicDarkColorScheme
import androidx.compose.material3.dynamicLightColorScheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.runtime.SideEffect
import androidx.compose.ui.graphics.toArgb
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.LocalView
import androidx.core.view.ViewCompat

private val DarkColorScheme = darkColorScheme(
    primary = Purple80,
    secondary = PurpleGrey80,
    tertiary = Pink80
)

private val LightColorScheme = lightColorScheme(
    primary = Purple40,
    secondary = PurpleGrey40,
    tertiary = Pink40

/* Other default colors to override
  background = Color(0xFFFFFBFE),
  surface = Color(0xFFFFFBFE),
  onPrimary = Color.White,
  onSecondary = Color.White,
  onTertiary = Color.White,
 onBackground = Color(0xFF1C1B1F),
  onSurface = Color(0xFF1C1B1F),
*/
)

@Composable
fun PythonPocketIDETheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme) dynamicDarkColorScheme(context) else dynamicLightColorScheme(context)
        }
        darkTheme -> DarkColorScheme
        else -> LightColorScheme
    }
    val view = LocalView.current
    if (!view.isInEditMode) {
        SideEffect {
            (view.context as Activity).window.statusBarColor = colorScheme.primary.toArgb()
            ViewCompat.getWindowInsetsController(view)?.isAppearanceLightStatusBars = darkTheme
        }
    }

    MaterialTheme(
        colorScheme = colorScheme,
        typography = Typography,
        content = content
    )
}
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/theme/Type.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/theme/Type.kt

File Type: Kotlin

Lines of Code: 34

```
package com.ankit.pythonpocketide.ui.theme

import androidx.compose.material3.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography = Typography(
    bodyLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp,
        lineHeight = 24.sp,
        letterSpacing = 0.5.sp
    )
    /* Other default text styles to override
    titleLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 22.sp,
        lineHeight = 28.sp,
        letterSpacing = 0.sp
    ),
    labelSmall = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Medium,
        fontSize = 11.sp,
        lineHeight = 16.sp,
        letterSpacing = 0.5.sp
    )
    */
)
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/theme/Color.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/theme/Color.kt

File Type: Kotlin

Lines of Code: 11

```
package com.ankit.pythonpocketide.ui.theme

import androidx.compose.ui.graphics.Color

    val Purple80 = Color(0xFFD0BCFF)
  val PurpleGrey80 = Color(0xFFCCC2DC)
     val Pink80 = Color(0xFFEFB8C8)

    val Purple40 = Color(0xFF6650a4)
  val PurpleGrey40 = Color(0xFF625b71)
     val Pink40 = Color(0xFF7D5260)
```

Kotlin File: app/src/main/java/com/ankit/pythonpocketide/ui/theme/EditorTheme.kt

File Path: app/src/main/java/com/ankit/pythonpocketide/ui/theme/EditorTheme.kt

File Type: Kotlin

Lines of Code: 9

```
package com.ankit.pythonpocketide.ui.theme

class EditorTheme {
    companion object {
        const val AbyssColor: String = "Abyss Color"
        const val DarculaTheme: String = "Dracula Theme"
        const val QuietLight: String = "Quiet Light"
    }
}
```

Java File: app/src/main/java/com/ankit/pythonpocketide/utils/CrashHandler.java

File Path: app/src/main/java/com/ankit/pythonpocketide/utils/CrashHandler.java

File Type: Java

Lines of Code: 148

```
package com.ankit.pythonpocketide.utils;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.os.Build;
import android.os.Handler;
import android.os.Looper;
import android.util.Log;

import androidx.annotation.NonNull;

import java.io.FileOutputStream;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.io.Writer;
import java.lang.Thread.UncaughtExceptionHandler;
import java.lang.reflect.Field;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;


/**
 * CrashHandler handles uncaught exceptions
 * And force the main thread continue to work
 *
 * @author Rosemoe
 */
public class CrashHandler implements UncaughtExceptionHandler {

    public final static String LOG_TAG = "CrashHandler";
    @SuppressLint("StaticFieldLeak")
    public final static CrashHandler INSTANCE = new CrashHandler();

    private Context mContext;
    private final Map<String, String> info = new HashMap<>();

    private CrashHandler() {
    }

    public void init(Context context) {
        mContext = context.getApplicationContext();
        collectDeviceInfo(mContext);
        Thread.setDefaultUncaughtExceptionHandler(this);
    }

    @Override
    public void uncaughtException(Thread thread, @NonNull Throwable ex) {
        saveCrashInfo(thread.getName(), ex);
        // Save the world, hopefully
        if (Looper.myLooper() != null) {
            Handler handler = new Handler(Looper.myLooper());
            while (true) {
                try {
                    handler.post(() -> {

                    });
                    Looper.loop();
                } catch (Throwable t) {
                    saveCrashInfo(thread.getName(), t);
                }
            }
        }
    }

    public void collectDeviceInfo(Context ctx) {
        try {
            PackageManager pm = ctx.getPackageManager();
            PackageInfo pi = pm.getPackageInfo(ctx.getPackageName(), PackageManager.GET_ACTIVITIES);
            if (pi != null) {
                String versionName = pi.versionName == null ? "null" : pi.versionName;
                String versionCode = pi.versionCode + "";
                info.put("versionName", versionName);
                info.put("versionCode", versionCode);
            }
        } catch (NameNotFoundException e) {
            Log.e(LOG_TAG, "an error occurred while collecting package info", e);
        }
        Field[] fields = Build.class.getDeclaredFields();
```

```
                                    for (Field field : fields) {
                                                try {
                                             field.setAccessible(true);
                                            Object obj = field.get(null);
                                           if (obj instanceof String[]) {
                                 info.put(field.getName(), Arrays.toString((String[]) obj));
                                                } else {
                                       info.put(field.getName(), String.valueOf(obj));
                                                }
                                             } catch (Exception e) {
                            Log.e(LOG_TAG, "an error occurred while collecting crash info", e);
                                                }
                                            }
                                 fields = Build.VERSION.class.getDeclaredFields();
                                            for (Field field : fields) {
                                                try {
                                             field.setAccessible(true);
                                            Object obj = field.get(null);
                                           if (obj instanceof String[]) {
                                 info.put(field.getName(), Arrays.toString((String[]) obj));
                                                } else {
                                       info.put(field.getName(), String.valueOf(obj));
                                                }
                                             } catch (Exception e) {
                            Log.e(LOG_TAG, "an error occurred while collecting crash info", e);
                                                }
                                            }
                                          }

                        private void saveCrashInfo(String threadName, Throwable ex) {
                                    StringBuilder sb = new StringBuilder();
                                  long timestamp = System.currentTimeMillis();
        sb.append("Crash at ").append(timestamp).append("(timestamp) in thread named '").append(threadNam
        sb.append("Local date and time:").append(SimpleDateFormat.getDateTimeInstance().format(new Date(t
                            for (Map.Entry<String, String> entry : info.entrySet()) {
                                       String key = entry.getKey();
                                     String value = entry.getValue();
                                sb.append(key).append("=").append(value).append("\n");
                                                }

                                        Writer writer = new StringWriter();
                                PrintWriter printWriter = new PrintWriter(writer);
                                         ex.printStackTrace(printWriter);
                                         Throwable cause = ex.getCause();
                                              while (cause != null) {
                                          cause.printStackTrace(printWriter);
                                               cause = cause.getCause();
                                                }
                                               printWriter.close();
                                        String result = writer.toString();
                                          sb.append(result).append('\n');
                                                    try {
                                             Log.e(LOG_TAG, sb.toString());
             FileOutputStream fos = mContext.openFileOutput("crash-journal.log", Context.MODE_APPEND);
                                         fos.write(sb.toString().getBytes());
                                                    fos.close();
                                             } catch (Exception e) {
                                Log.e(LOG_TAG, "an error occurred while writing file...", e);
                                                }
                                              }
                                            }
```

Kotlin File: app/src/androidTest/java/github/psicodes/ktxpy/ExampleInstrumentedTest.kt

File Path: app/src/androidTest/java/github/psicodes/ktxpy/ExampleInstrumentedTest.kt

File Type: Kotlin

Lines of Code: 24

```
package com.ankit.pythonpocketide

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.wildzeus.pythonktx", appContext.packageName)
    }
}
```

Kotlin File: app/src/test/java/github/psicodes/ktxpy/ExampleUnitTest.kt

File Path: app/src/test/java/github/psicodes/ktxpy/ExampleUnitTest.kt

File Type: Kotlin

Lines of Code: 17

```
package com.ankit.pythonpocketide

import org.junit.Test

import org.junit.Assert.*

/**
* Example local unit test, which will execute on the development machine (host).
*
* See [testing documentation](http://d.android.com/tools/testing).
*/
class ExampleUnitTest {
@Test
fun addition_isCorrect() {
assertEquals(4, 2 + 2)
}
}
```

XML File: app/src/main/res/values/styles.xml

File Path: app/src/main/res/values/styles.xml

File Type: XML

Lines of Code: 6

```
<resources>
    <style name="RobotoBoldTextAppearance">
        <item name="android:fontFamily">@font/roboto_condensed_bold</item>
    </style>
</resources>
```

XML File: app/src/main/res/values/themes.xml

File Path: app/src/main/res/values/themes.xml

File Type: XML

Lines of Code: 33

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="Theme.PythonKTX" parent="android:Theme.Material.Light.NoActionBar" />
    <style name="Theme.LayoutBasedActivities" parent="ThemeOverlay.Material3.DynamicColors.DayNight"/>
    <style name="AppTheme" parent="Theme.Material3.Dark.NoActionBar">
        <item name="colorPrimary">@color/md_theme_dark_primary</item>
        <item name="colorOnPrimary">@color/md_theme_dark_onPrimary</item>
        <item name="colorPrimaryContainer">@color/md_theme_dark_primaryContainer</item>
        <item name="colorOnPrimaryContainer">@color/md_theme_dark_onPrimaryContainer</item>
        <item name="colorSecondary">@color/md_theme_dark_secondary</item>
        <item name="colorOnSecondary">@color/md_theme_dark_onSecondary</item>
        <item name="colorSecondaryContainer">@color/md_theme_dark_secondaryContainer</item>
        <item name="colorOnSecondaryContainer">@color/md_theme_dark_onSecondaryContainer</item>
        <item name="colorTertiary">@color/md_theme_dark_tertiary</item>
        <item name="colorOnTertiary">@color/md_theme_dark_onTertiary</item>
        <item name="colorTertiaryContainer">@color/md_theme_dark_tertiaryContainer</item>
        <item name="colorOnTertiaryContainer">@color/md_theme_dark_onTertiaryContainer</item>
        <item name="colorError">@color/md_theme_dark_error</item>
        <item name="colorOnError">@color/md_theme_dark_onError</item>
        <item name="colorErrorContainer">@color/md_theme_dark_errorContainer</item>
        <item name="colorOnErrorContainer">@color/md_theme_dark_onErrorContainer</item>
        <item name="colorOutline">@color/md_theme_dark_outline</item>
        <item name="android:colorBackground">@color/md_theme_dark_background</item>
        <item name="colorOnBackground">@color/md_theme_dark_onBackground</item>
        <item name="colorSurface">@color/md_theme_dark_surface</item>
        <item name="colorOnSurface">@color/md_theme_dark_onSurface</item>
        <item name="colorSurfaceVariant">@color/md_theme_dark_surfaceVariant</item>
        <item name="colorOnSurfaceVariant">@color/md_theme_dark_onSurfaceVariant</item>
        <item name="colorSurfaceInverse">@color/md_theme_dark_inverseSurface</item>
        <item name="colorOnSurfaceInverse">@color/md_theme_dark_inverseOnSurface</item>
        <item name="colorPrimaryInverse">@color/md_theme_dark_inversePrimary</item>
    </style>
</resources>
```

XML File: app/src/main/res/values/strings.xml

File Path: app/src/main/res/values/strings.xml

File Type: XML

Lines of Code: 148

```
<resources>
    <string name="app_name">Python Pocket IDE</string>
    <string name="undo">Undo</string>
    <string name="redo">Redo</string>
    <string name="theme_select">Select Theme</string>
    <string name="save">Save File</string>
    <string name="add_file">Add File</string>
<string name="message_external_storage_rationale">Access to \"External Storage\" is required to manag
    <string name="run_code">Run Code</string>
    <string name="extra_menu">Extra Menu</string>

    <!-- PPIDE Enhanced Strings -->
<string name="app_description">Professional Python IDE for Android with AI-powered features</string>
    <string name="home">Home</string>
    <string name="projects">Projects</string>
    <string name="terminal">Terminal</string>
    <string name="libraries">Libraries</string>
    <string name="samples">Samples</string>
    <string name="editor">Editor</string>
    <string name="about">About</string>
    <string name="settings">Settings</string>

    <!-- Project Management -->
    <string name="create_project">Create Project</string>
    <string name="open_project">Open Project</string>
    <string name="delete_project">Delete Project</string>
    <string name="export_project">Export Project</string>
    <string name="import_project">Import Project</string>
    <string name="project_name">Project Name</string>
    <string name="project_description">Project Description</string>
    <string name="project_template">Project Template</string>
    <string name="project_files">Project Files</string>
    <string name="project_dependencies">Dependencies</string>
    <string name="project_settings">Project Settings</string>
    <string name="project_main_file">Main File</string>
    <string name="project_run">Run Project</string>
    <string name="project_terminal">Project Terminal</string>
    <string name="project_shell">Project Shell</string>

    <!-- Project Templates -->
    <string name="template_empty">Empty Project</string>
<string name="template_empty_desc">A minimal Python project structure</string>
    <string name="template_flask">Flask API</string>
<string name="template_flask_desc">A starter RESTful API using Flask framework</string>
    <string name="template_data_analysis">Data Analysis</string>
<string name="template_data_analysis_desc">Data analysis project with pandas and matplotlib</string>
    <string name="template_machine_learning">Machine Learning</string>
<string name="template_machine_learning_desc">ML project template with scikit-learn</string>

    <!-- Dependency Management -->
    <string name="install_dependencies">Install Dependencies</string>
    <string name="add_dependency">Add Dependency</string>
    <string name="remove_dependency">Remove Dependency</string>
    <string name="dependency_name">Package Name</string>
    <string name="dependency_version">Version</string>
    <string name="requirements_file">requirements.txt</string>
    <string name="pip_install">Pip Install</string>
    <string name="pip_list">Pip List</string>
    <string name="pip_upgrade">Pip Upgrade</string>

    <!-- File Management -->
    <string name="create_file">Create File</string>
    <string name="create_folder">Create Folder</string>
    <string name="rename_file">Rename File</string>
    <string name="delete_file">Delete File</string>
    <string name="copy_file">Copy File</string>
    <string name="move_file">Move File</string>
    <string name="file_explorer">File Explorer</string>
    <string name="python_files">Python Files</string>
    <string name="legacy_files">Legacy Files</string>
<string name="convert_to_project">Convert to Project</string>

    <!-- Code Execution -->
    <string name="run_file">Run File</string>
    <string name="run_project">Run Project</string>
    <string name="run_module">Run Module</string>
    <string name="debug_mode">Debug Mode</string>
    <string name="interactive_mode">Interactive Mode</string>
<string name="execution_completed">Execution completed</string>
    <string name="execution_failed">Execution failed</string>

    <!-- Editor Features -->
<string name="syntax_highlighting">Syntax Highlighting</string>
    <string name="code_completion">Code Completion</string>
    <string name="auto_indent">Auto Indent</string>
    <string name="line_numbers">Line Numbers</string>
```

```
    <string name="word_wrap">Word Wrap</string>
    <string name="search">Search</string>
    <string name="replace">Replace</string>
    <string name="goto_line">Go to Line</string>
    <string name="find_and_replace">Find and Replace</string>

    <!-- UI Actions -->
    <string name="create">Create</string>
    <string name="edit">Edit</string>
    <string name="delete">Delete</string>
    <string name="cancel">Cancel</string>
    <string name="confirm">Confirm</string>
    <string name="yes">Yes</string>
    <string name="no">No</string>
    <string name="ok">OK</string>
    <string name="done">Done</string>
    <string name="loading">Loading</string>
    <string name="error">Error</string>
    <string name="success">Success</string>
    <string name="warning">Warning</string>
    <string name="info">Info</string>

    <!-- Messages -->
    <string name="project_created_successfully">Project created successfully!</string>
    <string name="project_deleted_successfully">Project deleted successfully!</string>
    <string name="project_exported_successfully">Project exported successfully!</string>
    <string name="project_imported_successfully">Project imported successfully!</string>
    <string name="file_saved_successfully">File saved successfully!</string>
    <string name="dependencies_installed_successfully">Dependencies installed successfully!</string>
    <string name="invalid_project_name">Invalid project name</string>
    <string name="project_already_exists">Project already exists</string>
    <string name="no_projects_found">No projects found</string>
    <string name="no_files_found">No files found</string>
    <string name="permission_required">Permission required</string>
    <string name="storage_permission_required">Storage permission is required to access files</string>

    <!-- Help and Instructions -->
    <string name="getting_started">Getting Started</string>
    <string name="create_first_project">Create your first Python project to get started</string>
    <string name="project_help">Projects help you organize related Python files and manage dependencies</s
    <string name="template_help">Choose a template to get started quickly with common Python project type
    <string name="dependency_help">Manage Python packages required by your project</string>
    <string name="terminal_help">Use the terminal to run commands and interact with your Python environme

    <!-- Version and Credits -->
    <string name="version">Version</string>
    <string name="version_number">1.0.0</string>
    <string name="developer">Developer</string>
    <string name="developer_name">Ankit Kumar</string>
    <string name="university">Chandigarh University</string>
    <string name="course">MCA Final Year Project</string>
    <string name="based_on">Based on KtxPy by PsiCodes</string>
    <string name="open_source">Open Source Licenses</string>

    <!-- Feature Descriptions -->
    <string name="feature_project_management">Project-based development with templates</string>
    <string name="feature_dependency_management">Automatic dependency management with pip</string>
    <string name="feature_code_editor">Advanced code editor with syntax highlighting</string>
    <string name="feature_terminal">Integrated terminal with Python environment</string>
    <string name="feature_file_management">Complete file and folder management</string>
    <string name="feature_import_export">Project import/export capabilities</string>
</resources>
```

XML File: app/src/main/res/values/colors.xml

File Path: app/src/main/res/values/colors.xml

File Type: XML

Lines of Code: 73

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="colorvalue_alpha">#FFFFFFFF</color>
    <color name="colorvalue_red">#FFFF0000</color>
    <color name="colorvalue_green">#FF00FF00</color>
    <color name="colorvalue_blue">#FF0000FF</color>
    <color name="black">#000000</color>
    <color name="terminal_colour">#150E15</color>
    <color name="seed">#6750A4</color>
    <color name="md_theme_light_primary">#6750A4</color>
    <color name="md_theme_light_onPrimary">#FFFFFF</color>
    <color name="md_theme_light_primaryContainer">#EADDFF</color>
    <color name="md_theme_light_onPrimaryContainer">#21005D</color>
    <color name="md_theme_light_secondary">#625B71</color>
    <color name="md_theme_light_onSecondary">#FFFFFF</color>
    <color name="md_theme_light_secondaryContainer">#E8DEF8</color>
    <color name="md_theme_light_onSecondaryContainer">#1D192B</color>
    <color name="md_theme_light_tertiary">#7D5260</color>
    <color name="md_theme_light_onTertiary">#FFFFFF</color>
    <color name="md_theme_light_tertiaryContainer">#FFD8E4</color>
    <color name="md_theme_light_onTertiaryContainer">#31111D</color>
    <color name="md_theme_light_error">#B3261E</color>
    <color name="md_theme_light_onError">#FFFFFF</color>
    <color name="md_theme_light_errorContainer">#F9DEDC</color>
    <color name="md_theme_light_onErrorContainer">#410E0B</color>
    <color name="md_theme_light_outline">#79747E</color>
    <color name="md_theme_light_background">#FFFBFE</color>
    <color name="md_theme_light_onBackground">#1C1B1F</color>
    <color name="md_theme_light_surface">#FFFBFE</color>
    <color name="md_theme_light_onSurface">#1C1B1F</color>
    <color name="md_theme_light_surfaceVariant">#E7E0EC</color>
    <color name="md_theme_light_onSurfaceVariant">#49454F</color>
    <color name="md_theme_light_inverseSurface">#313033</color>
    <color name="md_theme_light_inverseOnSurface">#F4EFF4</color>
    <color name="md_theme_light_inversePrimary">#D0BCFF</color>
    <color name="md_theme_light_shadow">#000000</color>
    <color name="md_theme_light_surfaceTint">#6750A4</color>
    <color name="md_theme_light_outlineVariant">#CAC4D0</color>
    <color name="md_theme_light_scrim">#000000</color>
    <color name="md_theme_dark_primary">#D0BCFF</color>
    <color name="md_theme_dark_onPrimary">#381E72</color>
    <color name="md_theme_dark_primaryContainer">#4F378B</color>
    <color name="md_theme_dark_onPrimaryContainer">#EADDFF</color>
    <color name="md_theme_dark_secondary">#CCC2DC</color>
    <color name="md_theme_dark_onSecondary">#332D41</color>
    <color name="md_theme_dark_secondaryContainer">#4A4458</color>
    <color name="md_theme_dark_onSecondaryContainer">#E8DEF8</color>
    <color name="md_theme_dark_tertiary">#EFB8C8</color>
    <color name="md_theme_dark_onTertiary">#492532</color>
    <color name="md_theme_dark_tertiaryContainer">#633B48</color>
    <color name="md_theme_dark_onTertiaryContainer">#FFD8E4</color>
    <color name="md_theme_dark_error">#F2B8B5</color>
    <color name="md_theme_dark_onError">#601410</color>
    <color name="md_theme_dark_errorContainer">#8C1D18</color>
    <color name="md_theme_dark_onErrorContainer">#F9DEDC</color>
    <color name="md_theme_dark_outline">#938F99</color>
    <color name="md_theme_dark_background">#1C1B1F</color>
    <color name="md_theme_dark_onBackground">#E6E1E5</color>
    <color name="md_theme_dark_surface">#1C1B1F</color>
    <color name="md_theme_dark_onSurface">#E6E1E5</color>
    <color name="md_theme_dark_surfaceVariant">#49454F</color>
    <color name="md_theme_dark_onSurfaceVariant">#CAC4D0</color>
    <color name="md_theme_dark_inverseSurface">#E6E1E5</color>
    <color name="md_theme_dark_inverseOnSurface">#313033</color>
    <color name="md_theme_dark_inversePrimary">#6750A4</color>
    <color name="md_theme_dark_shadow">#000000</color>
    <color name="md_theme_dark_surfaceTint">#D0BCFF</color>
    <color name="md_theme_dark_outlineVariant">#49454F</color>
    <color name="md_theme_dark_scrim">#000000</color>
</resources>
```

XML File: app/src/main/res/values/attrs.xml

File Path: app/src/main/res/values/attrs.xml

File Type: XML

Lines of Code: 21

```
<?xml version="1.0" encoding="utf-8"?>
<!--
 * Copyright (C) 2018-2019 Roumen Petrov.  All rights reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 -->

<resources>
    <attr name="colorPrimaryLight" format="color"/>
    <attr name="themeTextColor" format="color"/>
</resources>
```

XML File: app/src/main/res/values/dimens.xml

File Path: app/src/main/res/values/dimens.xml

File Type: XML

Lines of Code: 10

```
<resources>
    <dimen name="text_margin">16dp</dimen>
    <dimen name="text2_margin">8dp</dimen>
    <dimen name="text4_margin">4dp</dimen>
    <dimen name="fab_margin">16dp</dimen>
<!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_margin">16dp</dimen>
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

XML File: app/src/main/res/values/ic_launcher_background.xml

File Path: app/src/main/res/values/ic_launcher_background.xml

File Type: XML

Lines of Code: 4

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="ic_launcher_background">#000000</color>
</resources>
```

XML File: app/src/main/res/menu/menu_main.xml

File Path: app/src/main/res/menu/menu_main.xml

File Type: XML

Lines of Code: 47

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto">
                                <item
                                    android:id="@+id/text_undo"
                                android:icon="@drawable/undo_icon"
                                   android:title="@string/undo"
                                   app:showAsAction="always" />
                                <item
                                    android:id="@+id/text_redo"
                                android:icon="@drawable/redo_icon"
                                   android:title="@string/redo"
                                   app:showAsAction="always" />
                                <item
                                  android:id="@+id/current_files"
                               android:icon="@drawable/current_files_icon"
                                android:title="@string/theme_select"
                                   app:showAsAction="always" />
                                <item
                                   android:id="@+id/extra_menu"
                                android:icon="@drawable/menu_icon"
                                     app:showAsAction="always"
                               android:title="@string/extra_menu">
                                    <menu>
                                       <item
                                        android:id="@+id/text_save"
                                  android:icon="@drawable/text_save_icon"
                                        android:title="@string/save"
                                        app:showAsAction="always" />
                                           <item
                                       android:id="@+id/text_file_add"
                                    android:icon="@drawable/add_file_icon"
                                       android:title="@string/add_file"
                                         app:showAsAction="always" />
                                           <item
                                         android:id="@+id/text_theme"
                                android:icon="@drawable/theme_selector_icon"
                                    android:title="@string/theme_select"
                                        app:showAsAction="always" />

                                          </menu>
                                        </item>

                                    </menu>

                                      <!--
                                      -->
```

XML File: app/src/main/res/xml/backup_rules.xml

File Path: app/src/main/res/xml/backup_rules.xml

File Type: XML

Lines of Code: 13

```
<?xml version="1.0" encoding="utf-8"?><!--
  Sample backup rules file; uncomment and customize as necessary.
  See https://developer.android.com/guide/topics/data/autobackup
  for details.
  Note: This file is ignored for devices older that API 31
See https://developer.android.com/about/versions/12/backup-restore
-->
<full-backup-content>
  <!--
  <include domain="sharedpref" path="."/>
  <exclude domain="sharedpref" path="device.xml"/>
-->
</full-backup-content>
```

XML File: app/src/main/res/xml/data_extraction_rules.xml

File Path: app/src/main/res/xml/data_extraction_rules.xml

File Type: XML

Lines of Code: 19

```
<?xml version="1.0" encoding="utf-8"?><!--
    Sample data extraction rules file; uncomment and customize as necessary.
See https://developer.android.com/about/versions/12/backup-restore#xml-changes
                                    for details.
                                       -->
                                <data-extraction-rules>
                                       <cloud-backup>
        <!-- TODO: Use <include> and <exclude> to control what is backed up.
                                         <include .../>
                                         <exclude .../>
                                              -->
                                       </cloud-backup>
                                            <!--
                                      <device-transfer>
                                         <include .../>
                                         <exclude .../>
                                      </device-transfer>
                                              -->
                                </data-extraction-rules>
```

XML File: app/src/main/res/drawable/app_icon.xml

File Path: app/src/main/res/drawable/app_icon.xml

File Type: XML

Lines of Code: 9

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="46"
    android:viewportHeight="46">
    <path
    android:fillColor="#bea2c8"
android:pathData="m18.95,30.85 l2.2,-2.2L16.5,24l4.6,-4.6 -2.2,-2.2 -6.8,6.8ZM29.05,30.85L35.9,24l-
    </vector>
```

XML File: app/src/main/res/drawable/python_icon.xml

File Path: app/src/main/res/drawable/python_icon.xml

File Type: XML

Lines of Code: 22

```
                    <vector android:height="63.75dp" android:viewportHeight="255"
                            android:viewportWidth="256" android:width="64dp"
xmlns:aapt="http://schemas.android.com/aapt" xmlns:android="http://schemas.android.com/apk/res/androi
<path android:pathData="M126.92,0.07C62.08,0.07 66.13,28.19 66.13,28.19L66.2,57.31L128.07,57.31L128.0
                                    <aapt:attr name="android:fillColor">
                                <gradient android:endX="155.06" android:endY="150.31"
                            android:startX="25.35" android:startY="23.2" android:type="linear">
                                    <item android:color="#FF387EB8" android:offset="0"/>
                                    <item android:color="#FF366994" android:offset="1"/>
                                                </gradient>
                                            </aapt:attr>
                                            </path>
<path android:pathData="M128.76,254.13C193.59,254.13 189.54,226.01 189.54,226.01L189.47,196.88L127.6,
                                    <aapt:attr name="android:fillColor">
                                <gradient android:endX="237.52" android:endY="231.9"
                            android:startX="98.21" android:startY="101.55" android:type="linear">
                                    <item android:color="#FFFFE052" android:offset="0"/>
                                    <item android:color="#FFFFC331" android:offset="1"/>
                                                </gradient>
                                            </aapt:attr>
                                            </path>
                                        </vector>
```

XML File: app/src/main/res/drawable/file_icon.xml

File Path: app/src/main/res/drawable/file_icon.xml

File Type: XML

Lines of Code: 5

```
                    <vector android:autoMirrored="true" android:height="24dp"
                         android:tint="#000000" android:viewportHeight="24"
android:viewportWidth="24" android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/and
<path android:fillColor="@android:color/white" android:pathData="M6,2c-1.1,0 -1.99,0.9 -1.99,2L4,20c0
                                        </vector>
```

XML File: app/src/main/res/drawable/about_icon.xml

File Path: app/src/main/res/drawable/about_icon.xml

File Type: XML

Lines of Code: 10

```
<vector
xmlns:android="http://schemas.android.com/apk/res/android"
android:width="24dp"
android:height="24dp"
android:viewportWidth="24.0"
android:viewportHeight="24.0">
<path
android:fillColor="#FFFFFFFF"
android:pathData="M11,18h2v-2h-2v2zM12,2C6.48,2 2,6.48 2,12s4.48,10 10,10 10,-4.48 10,-10S17.52,2
</vector>
```

XML File: app/src/main/res/drawable/theme_selector_icon.xml

File Path: app/src/main/res/drawable/theme_selector_icon.xml

File Type: XML

Lines of Code: 8

```
<vector android:width="24dp"
      android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
android:tint="?attr/colorControlNormal"
xmlns:android="http://schemas.android.com/apk/res/android">
<path android:fillColor="@android:color/white" android:pathData="M12,2C6.49,2 2,6.49 2,12s4.49,10 10,
</vector>
```

XML File: app/src/main/res/drawable/library_icon.xml

File Path: app/src/main/res/drawable/library_icon.xml

File Type: XML

Lines of Code: 5

```
                    <vector android:autoMirrored="true" android:height="24dp"
                         android:tint="#000000" android:viewportHeight="24"
android:viewportWidth="24" android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/and
<path android:fillColor="@android:color/white" android:pathData="M4,6L2,6v14c0,1.1 0.9,2 2,2h14v-2L4,
                                        </vector>
```

XML File: app/src/main/res/drawable/current_files_icon.xml

File Path: app/src/main/res/drawable/current_files_icon.xml

File Type: XML

Lines of Code: 6

```
                                <vector android:height="24dp"
                               android:tint="?attr/colorControlNormal"
                        android:viewportHeight="24" android:viewportWidth="24"
                android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
    <path android:fillColor="@android:color/white" android:pathData="M13,3c-4.97,0 -9,4.03 -9,9L1,12l3.89
                                        </vector>
```

XML File: app/src/main/res/drawable/coming_soon_icon.xml

File Path: app/src/main/res/drawable/coming_soon_icon.xml

File Type: XML

Lines of Code: 6

```
                        <vector android:height="24dp" android:tint="#000000"
                         android:viewportHeight="24" android:viewportWidth="24"
          android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
    <path android:fillColor="@android:color/white" android:pathData="M11.99,2C6.47,2 2,6.48 2,12s4.47,10 9
    <path android:fillColor="@android:color/white" android:pathData="M12.5,7H11v6l5.25,3.15 0.75,-1.23 -4
                                            </vector>
```

XML File: app/src/main/res/drawable/scrollbar_thumb_icon.xml

File Path: app/src/main/res/drawable/scrollbar_thumb_icon.xml

File Type: XML

Lines of Code: 11

```
<shape
xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle"
android:tint="@color/textActionNameColor">

<padding android:top="4dp" android:bottom="4dp" />

<size android:width="8dp" android:height="52dp" />

<solid android:color="@android:color/white" />
</shape>
```

XML File: app/src/main/res/drawable/app_icon_foreground.xml

File Path: app/src/main/res/drawable/app_icon_foreground.xml

File Type: XML

Lines of Code: 15

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="#CB99D3">
  <group android:scaleX="0.5858"
      android:scaleY="0.5858"
      android:translateX="4.9704"
      android:translateY="4.9704">
    <path
  android:fillColor="@android:color/white"
android:pathData="M9.4,16.6L4.8,12l4.6,-4.6L8,6l-6,6 6,6 1.4,-1.4zM14.6,16.6l4.6,-4.6 -4.6,-4.6L1(
    </group>
</vector>
```

XML File: app/src/main/res/drawable/back_icon.xml

File Path: app/src/main/res/drawable/back_icon.xml

File Type: XML

Lines of Code: 5

```
                    <vector android:autoMirrored="true" android:height="24dp"
                         android:tint="#000000" android:viewportHeight="24"
android:viewportWidth="24" android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/and
<path android:fillColor="@android:color/white" android:pathData="M20,11H7.83l5.59,-5.59L12,4l-8,8 8,8
                                            </vector>
```

XML File: app/src/main/res/drawable/folder_icon.xml

File Path: app/src/main/res/drawable/folder_icon.xml

File Type: XML

Lines of Code: 5

```
<vector android:height="24dp" android:tint="#000000"
 android:viewportHeight="24" android:viewportWidth="24"
android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
<path android:fillColor="@android:color/white" android:pathData="M10,4H4c-1.1,0 -1.99,0.9 -1.99,2L2,18
</vector>
```

XML File: app/src/main/res/drawable/text_save_icon.xml

File Path: app/src/main/res/drawable/text_save_icon.xml

File Type: XML

Lines of Code: 9

```
<vector
android:width="24dp"
android:height="24dp"
android:viewportWidth="24"
android:viewportHeight="24"
android:tint="?attr/colorControlNormal"
xmlns:android="http://schemas.android.com/apk/res/android">
<path android:fillColor="@android:color/white" android:pathData="M17,3L5,3c-1.11,0 -2,0.9 -2,2v14c0,1
</vector>
```

XML File: app/src/main/res/drawable/interactive_mode_icon.xml

File Path: app/src/main/res/drawable/interactive_mode_icon.xml

File Type: XML

Lines of Code: 6

```
                        <vector android:height="24dp" android:tint="#000000"
                         android:viewportHeight="24" android:viewportWidth="24"
                android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
    <path android:fillColor="@android:color/white" android:pathData="M6.41,6l-1.41,1.41l4.58,4.59l-4.58,4
    <path android:fillColor="@android:color/white" android:pathData="M13,6l-1.41,1.41l4.58,4.59l-4.58,4.5
                                            </vector>
```

XML File: app/src/main/res/drawable/code_run_icon.xml

File Path: app/src/main/res/drawable/code_run_icon.xml

File Type: XML

Lines of Code: 5

```
<vector android:height="24dp" android:tint="#000000"
 android:viewportHeight="24" android:viewportWidth="24"
 android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
<path android:fillColor="@android:color/white" android:pathData="M8,5v14l11,-7z"/>
</vector>
```

XML File: app/src/main/res/drawable/terminal_icon.xml

File Path: app/src/main/res/drawable/terminal_icon.xml

File Type: XML

Lines of Code: 5

```
                        <vector android:height="24dp" android:tint="#000000"
                         android:viewportHeight="24" android:viewportWidth="24"
                android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
    <path android:fillColor="@android:color/white" android:pathData="M20,4H4C2.89,4 2,4.9 2,6v12c0,1.1 0.8
                                                </vector>
```

XML File: app/src/main/res/drawable/file_open_icon.xml

File Path: app/src/main/res/drawable/file_open_icon.xml

File Type: XML

Lines of Code: 5

```
<vector android:height="24dp" android:tint="#000000"
 android:viewportHeight="24" android:viewportWidth="24"
android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
<path android:fillColor="@android:color/white" android:pathData="M14,2H6C4.9,2 4,2.9 4,4v16c0,1.1 0.8
</vector>
```

XML File: app/src/main/res/drawable/white_cursor_icon.xml

File Path: app/src/main/res/drawable/white_cursor_icon.xml

File Type: XML

Lines of Code: 6

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#FFFFFF" />
    <size android:width="2dp" />
</shape>
```

XML File: app/src/main/res/drawable/sample_icon.xml

File Path: app/src/main/res/drawable/sample_icon.xml

File Type: XML

Lines of Code: 5

```
                        <vector android:height="24dp" android:tint="#000000"
                         android:viewportHeight="24" android:viewportWidth="24"
          android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
  <path android:fillColor="@android:color/white" android:pathData="M20,18c1.1,0 2,-0.9 2,-2V6c0,-1.1 -0
                                                  </vector>
```

XML File: app/src/main/res/drawable/add_file_icon.xml

File Path: app/src/main/res/drawable/add_file_icon.xml

File Type: XML

Lines of Code: 5

```
<vector android:autoMirrored="true" android:height="24dp"
android:tint="?attr/colorControlNormal" android:viewportHeight="24"
android:viewportWidth="24" android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/and
<path android:fillColor="@android:color/white" android:pathData="M14,2L6,2c-1.1,0 -1.99,0.9 -1.99,2L4
</vector>
```

XML File: app/src/main/res/drawable/create_file_icon.xml

File Path: app/src/main/res/drawable/create_file_icon.xml

File Type: XML

Lines of Code: 5

```
<vector android:height="24dp" android:tint="#000000"
 android:viewportHeight="24" android:viewportWidth="24"
android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
<path android:fillColor="@android:color/white" android:pathData="M20,6h-8l-2,-2L4,4c-1.11,0 -1.99,0.8
</vector>
```

XML File: app/src/main/res/drawable/menu_icon.xml

File Path: app/src/main/res/drawable/menu_icon.xml

File Type: XML

Lines of Code: 6

```
                                <vector android:height="24dp"
                                android:tint="?attr/colorControlNormal"
                            android:viewportHeight="24" android:viewportWidth="24"
                    android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">
    <path android:fillColor="@android:color/white" android:pathData="M3,18h18v-2L3,16v2zM3,13h18v-2L3,11v
                                            </vector>
```

XML File: app/src/main/res/layout/activity_editor.xml

File Path: app/src/main/res/layout/activity_editor.xml

File Type: XML

Lines of Code: 64

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/andr
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.google.android.material.appbar.MaterialToolbar
        android:id="@+id/materialToolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@id/editor"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:id="@+id/run_code"
        android:layout_margin="16dp"
        app:layout_constraintBottom_toTopOf="@+id/navigation_keyboard_scroll"
        app:layout_constraintEnd_toEndOf="parent"
        android:contentDescription="@string/run_code" />

    <io.github.rosemoe.sora.widget.CodeEditor
        android:id="@+id/editor"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:scrollbarThumbHorizontal="@drawable/scrollbar_thumb_icon"
        android:scrollbarThumbVertical="@drawable/scrollbar_thumb_icon"
        android:scrollbarTrackHorizontal="@drawable/scrollbar_track_icon"
        android:scrollbarTrackVertical="@drawable/scrollbar_track_icon"
        app:layout_constraintBottom_toTopOf="@id/navigation_keyboard_scroll"
        app:layout_constraintTop_toBottomOf="@id/materialToolbar"
        app:layout_constraintVertical_bias="0.0"
        app:lnPanelPosition="center"
        app:lnPanelPositionMode="follow"
        tools:layout_editor_absoluteX="0dp" />

    <include
        android:id="@+id/navigation_keyboard_scroll"
        layout="@layout/keyboard_navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@id/symbol_input_scroll" />

    <HorizontalScrollView
        android:id="@+id/symbol_input_scroll"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        app:layout_constraintBottom_toBottomOf="parent">

        <io.github.rosemoe.sora.widget.SymbolInputView
            android:id="@+id/symbol_input"
            android:layout_width="wrap_content"
            android:layout_height="match_parent" />

    </HorizontalScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

XML File: app/src/main/res/layout/keyboard_terminal.xml

File Path: app/src/main/res/layout/keyboard_terminal.xml

File Type: XML

Lines of Code: 150

```
<?xml version="1.0" encoding="utf-8"?>
<HorizontalScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/terminal_keyboard_scroll"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#1A1A1A">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="48dp"
        android:gravity="center"
        android:orientation="horizontal">

        <Button
            android:id="@+id/key_tab"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="Tab"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="12sp"/>

        <Button
            android:id="@+id/key_esc"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="Esc"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="12sp"/>

        <Button
            android:id="@+id/key_ctrl"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="Ctrl"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="12sp"/>

        <Button
            android:id="@+id/key_alt"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="Alt"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="12sp"/>

        <Button
            android:id="@+id/key_left"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="←"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="18sp"/>

        <Button
            android:id="@+id/key_right"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="→"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="18sp"/>

        <Button
            android:id="@+id/key_up"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="↑"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="18sp"/>

        <Button
            android:id="@+id/key_down"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="↓"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="18sp"/>

        <Button
```

```
            android:id="@+id/key_pipe"
            android:layout_width="48dp"
        android:layout_height="match_parent"
                android:text="|"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="18sp"/>

                    <Button
            android:id="@+id/key_minus"
            android:layout_width="48dp"
        android:layout_height="match_parent"
                android:text="-"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="18sp"/>

                    <Button
            android:id="@+id/key_home"
            android:layout_width="48dp"
        android:layout_height="match_parent"
               android:text="Home"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="10sp"/>

                    <Button
            android:id="@+id/key_end"
            android:layout_width="48dp"
        android:layout_height="match_parent"
                android:text="End"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="10sp"/>

                    <Button
            android:id="@+id/key_pgup"
            android:layout_width="48dp"
        android:layout_height="match_parent"
               android:text="PgUp"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="10sp"/>

                    <Button
            android:id="@+id/key_pgdn"
            android:layout_width="48dp"
        android:layout_height="match_parent"
               android:text="PgDn"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="10sp"/>

                    <Button
       android:id="@+id/key_toggle_keyboard"
            android:layout_width="48dp"
        android:layout_height="match_parent"
                android:text="⌨"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="18sp"/>
              </LinearLayout>
        </HorizontalScrollView>
```

XML File: app/src/main/res/layout/activity_terminal.xml

File Path: app/src/main/res/layout/activity_terminal.xml

File Type: XML

Lines of Code: 32

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/andr
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/terminal_colour">

    <com.termux.view.TerminalView
        android:id="@+id/terminal_view"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@id/terminal_keyboard_container" />

    <LinearLayout
        android:id="@+id/terminal_keyboard_container"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent">

        <include
            android:id="@+id/terminal_keyboard_scroll"
            layout="@layout/keyboard_terminal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

XML File: app/src/main/res/layout/keyboard_navigation.xml

File Path: app/src/main/res/layout/keyboard_navigation.xml

File Type: XML

Lines of Code: 132

```
<?xml version="1.0" encoding="utf-8"?>
<HorizontalScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/navigation_keyboard_scroll"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#1E1E1E">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="48dp"
        android:gravity="center"
        android:orientation="horizontal">

        <Button
            android:id="@+id/key_tab"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="Tab"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="12sp"/>

        <Button
            android:id="@+id/key_esc"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="Esc"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="12sp"/>

        <Button
            android:id="@+id/key_ctrl"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="Ctrl"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="12sp"/>

        <Button
            android:id="@+id/key_left"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="←"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="18sp"/>

        <Button
            android:id="@+id/key_right"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="→"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="18sp"/>

        <Button
            android:id="@+id/key_up"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="↑"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="18sp"/>

        <Button
            android:id="@+id/key_down"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="↓"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="18sp"/>

        <Button
            android:id="@+id/key_home"
            android:layout_width="48dp"
            android:layout_height="match_parent"
            android:text="Home"
            style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
            android:textSize="10sp"/>

        <Button
```

```
            android:id="@+id/key_end"
            android:layout_width="48dp"
        android:layout_height="match_parent"
                android:text="End"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="10sp"/>

                    <Button
             android:id="@+id/key_pgup"
            android:layout_width="48dp"
        android:layout_height="match_parent"
                android:text="PgUp"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="10sp"/>

                    <Button
             android:id="@+id/key_pgdn"
            android:layout_width="48dp"
        android:layout_height="match_parent"
                android:text="PgDn"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="10sp"/>

                    <Button
             android:id="@+id/key_del"
            android:layout_width="48dp"
        android:layout_height="match_parent"
                android:text="Del"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="12sp"/>

                    <Button
       android:id="@+id/key_toggle_keyboard"
            android:layout_width="48dp"
       android:layout_height="match_parent"
                 android:text="⌨"
    style="?android:attr/buttonBarButtonStyle"
            android:textColor="#FFFFFF"
             android:textSize="18sp"/>
              </LinearLayout>
       </HorizontalScrollView>
```

XML File: app/src/main/res/drawable-v24/redo_icon.xml

File Path: app/src/main/res/drawable-v24/redo_icon.xml

File Type: XML

Lines of Code: 35

```
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                ~    sora-editor - the awesome code editor for Android
                 ~    https://github.com/Rosemoe/sora-editor
                    ~    Copyright (C) 2020-2022  Rosemoe
                                        ~
          ~     This library is free software; you can redistribute it and/or
           ~     modify it under the terms of the GNU Lesser General Public
          ~     License as published by the Free Software Foundation; either
       ~     version 2.1 of the License, or (at your option) any later version.
                                        ~
          ~     This library is distributed in the hope that it will be useful,
          ~     but WITHOUT ANY WARRANTY; without even the implied warranty of
       ~     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
                 ~     Lesser General Public License for more details.
                                        ~
          ~     You should have received a copy of the GNU Lesser General Public
       ~     License along with this library; if not, write to the Free Software
    ~     Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301
                                     ~     USA
                                        ~
          ~     Please contact Rosemoe by email 2073412493@qq.com if you need
                 ~     additional information or have any questions
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->

          <vector xmlns:android="http://schemas.android.com/apk/res/android"
                                  android:width="24dp"
                                 android:height="24dp"
                                android:viewportWidth="24"
                                android:viewportHeight="24"
                             android:tint="?attr/colorControlNormal"
                                  android:autoMirrored="true">
                                        <path
                                 android:fillColor="@android:color/white"
android:pathData="M18.4,10.6C16.55,8.99 14.15,8 11.5,8c-4.65,0 -8.58,3.03 -9.96,7.22L3.9,16c1.05,-3
                                        </vector>
```

XML File: app/src/main/res/drawable-v24/scrollbar_track_icon.xml

File Path: app/src/main/res/drawable-v24/scrollbar_track_icon.xml

File Type: XML

Lines of Code: 9

```
<shape
xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle"
android:tint="?colorControlNormal">

<size android:width="8dp" />

<solid android:color="#39FFFFFF" />
</shape>
```

XML File: app/src/main/res/drawable-v24/undo_icon.xml

File Path: app/src/main/res/drawable-v24/undo_icon.xml

File Type: XML

Lines of Code: 11

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="?attr/colorControlNormal"
    android:autoMirrored="true">
    <path
    android:fillColor="@android:color/white"
android:pathData="M12.5,8c-2.65,0 -5.05,0.99 -6.9,2.6L2,7v9h9l-3.62,-3.62c1.39,-1.16 3.16,-1.88 5.1
    </vector>
```

XML File: app/src/main/res/mipmap/ic_launcher_round.xml

File Path: app/src/main/res/mipmap/ic_launcher_round.xml

File Type: XML

Lines of Code: 6

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@color/ic_launcher_background"/>
    <foreground android:drawable="@drawable/app_icon_foreground"/>
    <monochrome android:drawable="@drawable/app_icon_foreground"/>
</adaptive-icon>
```

XML File: app/src/main/res/mipmap/ic_launcher.xml

File Path: app/src/main/res/mipmap/ic_launcher.xml

File Type: XML

Lines of Code: 6

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@color/ic_launcher_background"/>
    <foreground android:drawable="@drawable/app_icon_foreground"/>
    <monochrome android:drawable="@drawable/app_icon_foreground"/>
</adaptive-icon>
```

Error reading file /home/ankit/Desktop/ktxpy/PythonPocketIDE_FYP/PPIDE_Source/.gradle: [Errno 21] Is a directory: '/home/ankit/Desktop/ktxpy/PythonPocketIDE_FYP/PPIDE_Source/.gradle'

Gradle File: build.gradle

File Path: build.gradle

File Type: Gradle

Lines of Code: 17

```
buildscript {
    ext {
        compose_version = '1.1.1'
    }
    project.ext{
        compileSdkVersion = 32
        targetSdkVersion = 33
        minSdkVersion = 26
        ndkVersion = '25.1.8937393'
    }
}// Top-level build file where you can add configuration options common to all sub-projects/modules.
plugins {
    id 'com.android.application' version '8.1.2' apply false
    id 'com.android.library' version '8.1.2' apply false
    id 'org.jetbrains.kotlin.android' version '1.9.10' apply false
    id 'com.google.dagger.hilt.android' version '2.44' apply false
}
```

Gradle File: settings.gradle

File Path: settings.gradle

File Type: Gradle

Lines of Code: 20

```
pluginManagement {
    repositories {
       gradlePluginPortal()
            google()
        mavenCentral()
maven { url 'https://jitpack.io' }
        }
    }
dependencyResolutionManagement {
repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
maven { url 'https://jitpack.io' }

    }
}
rootProject.name = "PythonPocketIDE"
include ':app'
include ':libp7zip'
```

Gradle File: libp7zip/build.gradle

File Path: libp7zip/build.gradle

File Type: Gradle

Lines of Code: 48

```
apply plugin: 'com.android.library'

android {
compileSdk 34

defaultConfig {
  minSdkVersion 26
 targetSdkVersion 34
externalNativeBuild {
cmake {
arguments '-DANDROID_STL=c++_static', '-DANDROID_PLATFORM=android-18'
}
}
}
buildTypes {
debug {
externalNativeBuild {
cmake {
// log switch
cppFlags.add('-DNATIVE_LOG')
}
}
}
}
externalNativeBuild {
cmake {
path 'src/main/cpp/CMakeLists.txt'
}
}
ndkVersion '25.1.8937393'
namespace 'com.hzy.libp7zip'
lint {
abortOnError false
}
}

dependencies {
implementation fileTree(dir: 'libs', include: ['*.jar'])
}

//publish {
//    userOrg = 'huzongyao'
//    groupId = 'com.hzy'
//    artifactId = 'libp7zip'
//    publishVersion = '1.7.1'
//    desc = 'An Android compress and extract library support popular compression format such as rar, zip
//    website = 'https://github.com/hzy3774/AndroidP7zip'
//}
```

Gradle File: app/build.gradle

File Path: app/build.gradle

File Type: Gradle

Lines of Code: 170

```
                                    plugins {
                                id 'com.android.application'
                               id 'org.jetbrains.kotlin.android'
                    id 'com.google.devtools.ksp' version '1.9.10-1.0.13'
                                id 'kotlin-parcelize'
                                        }

                                    android {
                            namespace 'com.ankit.pythonpocketide'
                                     compileSdk 34
                                  ndkVersion '25.1.8937393'
                                      defaultConfig {
                            applicationId "com.ankit.pythonpocketide"
                                         minSdk 26
                                        targetSdk 34
                                     versionName "2.0.0"
                                       versionCode 2
                        android.packagingOptions.jniLibs.useLegacyPackaging true
                                    externalNativeBuild {
                                            cmake {
                                arguments "-DAPPLICATION_ID:STRING=${applicationId}"
                                                }
                                                }
                               testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
                                         vectorDrawables {
                                        useSupportLibrary true
                                                }
                                            }
                                flavorDimensions += "cpuArch"
                                       productFlavors {
                                           arch_arm32 {
                                          dimension 'cpuArch'
                                          versionCode 4*1000+2
                                        versionNameSuffix "-arm32"
                                                }
                                           arch_arm64 {
                                          dimension 'cpuArch'
                                          versionCode 3*1000+2
                                        versionNameSuffix "-arm64"
                                                }
                                            arch_x86 {
                                          dimension 'cpuArch'
                                          versionCode 2*1000+2
                                         versionNameSuffix "-x86"
                                                }
                                           arch_x86_64 {
                                          dimension 'cpuArch'
                                          versionCode 1*1000+2
                                        versionNameSuffix "-x86_64"
                                                }
                                            }
                            buildFeatures.viewBinding = true
                                             buildTypes {
                                                release {
                                             minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pr
                                    signingConfig signingConfigs.debug
                                                }
                                            }
                                         compileOptions {
                                sourceCompatibility JavaVersion.VERSION_17
                                targetCompatibility JavaVersion.VERSION_17
                                                }
                                            kotlin {
                                            sourceSets {
                                                 debug{
                             kotlin.srcDir("build/generated/ksp/debug/kotlin")
                                                 }
                                               release{
                            kotlin.srcDir("build/generated/ksp/release/kotlin")
                                                 }
                                            }
                                        }
                                        kotlinOptions {
                                       freeCompilerArgs += [
                                            "-Xjvm-default=all"
                                                 ]
                                                }
                                       composeOptions {
                             kotlinCompilerExtensionVersion = "1.5.3"
                                                }
                                         buildFeatures {
                                             compose true
                                                }
                                           splits{
                                              abi {
```

```
                                        enable true
                                          reset()
                    include 'x86', 'x86_64', 'armeabi-v7a', 'arm64-v8a'
                                    universalApk false
                                          }
                                        }
                                    sourceSets {

                                      arch_arm64{
                                        sourceSets {
                                assets.srcDirs = ["arch_arm64-v8a/assets"]
                                          }
                                        }
                                      arch_arm32 {
                                        sourceSets {
                                  assets.srcDirs = ["arch_arm32/assets"]
                                          }
                                        }
                                       arch_x86 {
                                        sourceSets {
                                   assets.srcDirs = ["arch_x86/assets"]
                                          }
                                        }
                                      arch_x86_64 {
                                        sourceSets {
                                  assets.srcDirs = ["arch_x86_64/assets"]
                                          }
                                        }
                                      }
                                  packagingOptions {
                                        jniLibs {
                                     useLegacyPackaging true
                                          }
                                       }
                                     }
                                  dependencies {
            implementation fileTree(dir: 'libs', include: ['*.*'])
                                   //Android Related
                 implementation('androidx.core:core-ktx:1.12.0')
              implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.6.2")
               implementation "androidx.datastore:datastore-preferences:1.0.0"
            implementation "androidx.datastore:datastore-preferences-core:1.0.0"
                 implementation('androidx.activity:activity-compose:1.8.0')
            implementation(platform('androidx.compose:compose-bom:2023.10.00'))
                        implementation("androidx.compose.ui:ui")
                implementation("androidx.compose.runtime:runtime-livedata")
                     implementation("androidx.compose.ui:ui-graphics")
                implementation("androidx.compose.ui:ui-tooling-preview")
                 implementation("androidx.compose.material3:material3")
            implementation 'androidx.coordinatorlayout:coordinatorlayout:1.2.0'
                implementation 'com.google.android.material:material:1.10.0'
                    implementation 'androidx.appcompat:appcompat:1.6.1'
            implementation ('androidx.constraintlayout:constraintlayout:2.1.4')
                  debugImplementation("androidx.compose.ui:ui-tooling")
                implementation("androidx.compose.ui:ui-tooling-preview")

                           // lib AndroidUtilCode by  BlankJ
                     implementation("com.blankj:utilcodex:1.31.1")

                   // lib terminal-view and terminal-editor by Termux
             implementation("com.github.termux.termux-app:terminal-view:v0.118.0")
           implementation("com.github.termux.termux-app:terminal-emulator:v0.118.0")

                               // lib : p7zip by Hzy3774
                           implementation project(':libp7zip')

                             // lib : Sora-editor by Rosemoe
                    implementation 'io.github.Rosemoe.sora-editor:editor'
             implementation(platform('io.github.Rosemoe.sora-editor:bom:0.22.1'))
               implementation("io.github.Rosemoe.sora-editor:language-textmate")

                         // lib : Compose-Destination by RaamCosta
        implementation 'io.github.raamcosta.compose-destinations:animations-core:1.9.54'
                    ksp 'io.github.raamcosta.compose-destinations:ksp:1.9.54'

                                        // Test
                           testImplementation 'junit:junit:4.13.2'
                 androidTestImplementation 'androidx.test.ext:junit:1.1.5'
            androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
             androidTestImplementation 'androidx.compose.ui:ui-test-junit4:1.5.4'
               debugImplementation 'androidx.compose.ui:ui-test-manifest:1.5.4'
             debugImplementation 'com.squareup.leakcanary:leakcanary-android:2.12'
                                        }
```

Manifest File: app/src/main/AndroidManifest.xml

File Path: app/src/main/AndroidManifest.xml

File Type: Manifest

Lines of Code: 49

```
                <?xml version="1.0" encoding="utf-8"?>
  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools">

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
 <uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE"
                      tools:ignore="ScopedStorage" />
       <uses-permission android:name="android.permission.INTERNET"/>
                           <application
             android:name="com.ankit.pythonpocketide.Application"
                         android:allowBackup="true"
          android:dataExtractionRules="@xml/data_extraction_rules"
               android:fullBackupContent="@xml/backup_rules"
                    android:icon="@mipmap/ic_launcher"
                     android:label="@string/app_name"
                        android:supportsRtl="true"
                 android:requestLegacyExternalStorage="true"
                        tools:targetApi="34">
                              <activity
        android:name="com.ankit.pythonpocketide.activities.TermActivity"
                    android:theme="@style/Theme.PythonKTX"
                            android:exported="false"
                  android:windowSoftInputMode="adjustResize"
                                    />
                              <activity
        android:name="com.ankit.pythonpocketide.activities.HomeActivity"
                             android:exported="true"
                   android:theme="@style/Theme.PythonKTX">
                                <intent-filter>
              <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
                                </intent-filter>
                                </activity>
                                 <activity
       android:name="com.ankit.pythonpocketide.activities.EditorActivity"
                           android:theme="@style/AppTheme"
                  android:windowSoftInputMode="adjustResize"
                            android:exported="true">
                                 <intent-filter>
              <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT" />
                     <data android:mimeType="text/x-python" />
                         <data android:scheme="content" />
                           <data android:scheme="file" />
                                </intent-filter>
                                </activity>
                             </application>

                              </manifest>
```