***[KPTKP - BahtBuddy]***

**Software Deployment Plan**

***[Latest update: 2025-10-20*** *(keep updated when you make changes)****]***

**1. System Requirements**

For the BahtBuddy system, the application is designed to operate across multiple operating systems to ensure accessibility and usability for a wide range of users. The software supports Windows 10 and above, macOS (versions 12 Monterey and 13 Ventura), and Linux distributions including Ubuntu 20.04 and later, as well as openSUSE Leap 15. This cross-platform compatibility ensures that both personal and institutional users can install and use the program without being limited by their operating system choice. To guarantee consistent performance across all supported environments, the application will be tested on each platform during the deployment phase to confirm stable operation, proper graphical rendering through the Tkinter interface, and reliable data management via the embedded SQLite database.

In terms of hardware requirements, BahtBuddy is optimized to run efficiently on standard consumer hardware. The minimum recommended system specification includes a dual-core processor (1.6 GHz or higher), 4 GB of RAM, and at least 200 MB of available storage space. These requirements make it suitable for use on laptops, desktops, and low-power systems. However, for smoother performance—especially when managing large financial datasets or generating graphical reports—a system with 8 GB of RAM and an SSD is recommended.

Regarding software dependencies, BahtBuddy is built primarily in Python 3.10 or higher, relying on core libraries such as tkinter for the graphical user interface, sqlite3 for database handling. The only prerequisite for installation is a functioning Python environment with the package manager pip, which is typically included by default in most Python distributions. All other dependencies are handled automatically through the deployment package or installation script, ensuring that end users do not need to manually configure any components.

**2. Deployment Strategy Summary**

The deployment strategy for BahtBuddy focuses on delivering a seamless installation experience for end users while ensuring the application remains lightweight, portable, and easy to maintain. The general approach is to package the software as a standalone desktop application that requires no manual configuration or external dependencies beyond the included setup files. This ensures that users can install and run BahtBuddy without needing prior technical knowledge or administrative privileges, which aligns with the project’s goal of accessibility for individuals managing personal finances.

To achieve this, the deployment process will utilize PyInstaller, a Python packaging tool that converts Python scripts into executable files compatible with different operating systems. This tool will be used to generate a Windows executable (.exe), a macOS application bundle (.app), and a Linux binary, each containing the Python interpreter, required libraries, and the BahtBuddy source code. This approach removes the need for users to install Python or additional modules manually. For the Windows environment, an optional Inno Setup installer will be used to create a guided installation wizard, which places application shortcuts on the desktop and start menu while managing uninstall procedures automatically.

The final distribution kit will include the compiled application, necessary configuration files, database templates, graphical assets, and user documentation. Distribution will be handled through GitHub Releases, allowing users to easily download the appropriate version for their operating system. Each release will also contain a README file outlining system requirements, installation steps, and troubleshooting instructions. In future iterations, BahtBuddy may adopt an auto-update mechanism or integrate with a cloud-based deployment service to simplify version management and user updates. Overall, this deployment strategy ensures that BahtBuddy remains simple to install, platform-independent, and maintainable throughout its lifecycle.

**3. Installation Package Contents**

The installation package for BahtBuddy contains all essential components required for the system to operate as a self-contained personal finance management application. The folder structure is organized for clarity and modularity, ensuring that both end users and developers can easily identify the purpose of each file. Each component within the package contributes to a specific aspect of system functionality, from data processing to interface rendering and validation.

The required source files include several Python scripts that collectively define the program’s logic. The central file, main.py, serves as the entry point for the BahtBuddy application, initializing the graphical interface and managing user interactions. The gui.py module implements the Tkinter-based interface, defining windows, input fields, and buttons used throughout the program. The database.py module handles all database operations using SQLite, such as storing transactions, retrieving records, and managing user data. The validation.py script ensures that all user inputs—such as monetary values, dates, and account names—are verified for correctness before processing. These Python files are compiled and packaged together during deployment, forming the executable core of the system.

In addition to the main source files, the installation package includes supporting and configuration files. The .gitignore files are used to exclude unnecessary development or cache data from version control and packaging, maintaining a clean and efficient deployment. The LICENSE file provides the legal framework for distribution and defines user rights and restrictions associated with the software. The README.md file offers essential setup guidance, detailing installation steps, usage instructions, and an overview of BahtBuddy’s key features. These files enhance both usability and maintainability by providing clear documentation and compliance information within the distribution kit.

During deployment, these files will be bundled into an executable application using PyInstaller, ensuring that dependencies such as Tkinter and SQLite are included automatically. The resulting build will allow users to run BahtBuddy without needing to install Python or configure additional packages manually. Overall, this structured and modular installation package ensures that the system is lightweight, well-documented, and ready for reliable deployment across multiple operating systems.

***3.1 Required source or compiled files***

The required source files for the BahtBuddy system consist of the core Python scripts that collectively define the functional and operational behavior of the application. Each file serves a distinct purpose, contributing to the modular architecture that ensures maintainability, scalability, and clear separation of concerns within the codebase. Together, these components form the executable foundation of the program when compiled for deployment.

The central file, main.py, acts as the primary entry point for the entire system. It initializes the program, manages the launch sequence, and connects the graphical user interface (GUI) to the backend logic. This script ensures that all necessary modules are properly imported and that the application runs smoothly when executed.

The gui.py file handles the visual presentation and user interaction layer. Built using the Tkinter framework, this script defines the structure of all windows, input forms, buttons, and menus within the program. It also manages event-driven interactions, such as user input validation, menu navigation, and the triggering of backend functions in response to user actions.

The database.py module manages all database-related operations using SQLite. It is responsible for creating, reading, updating, and deleting records from the local database file that stores user financial information, such as transactions, budgets, and account balances. This script ensures data integrity, efficient queries, and smooth interaction between the front-end interface and the storage system.

The validation.py file serves as a data integrity layer, verifying that all user inputs are valid before being stored or processed. This includes checking for missing values, incorrect data formats, or invalid transaction amounts. By isolating validation logic into a dedicated module, the system maintains reliability and reduces the likelihood of data-related errors.

Finally, the supporting files in the package—such as .gitignore, LICENSE, and README.md—play essential roles during development and deployment. The .gitignore file prevents unnecessary system or cache files from being tracked in version control, ensuring a clean repository. The LICENSE file outlines the legal permissions and restrictions for software distribution, while the README.md provides installation guidance, an overview of features, and basic troubleshooting steps.

In the compiled version of BahtBuddy, these files are packaged together using PyInstaller into a single executable file. This approach encapsulates the entire system into a portable, ready-to-run distribution suitable for Windows, macOS, and Linux environments.

***3.2 Required third-party components***

The third-party components required for the BahtBuddy application are primarily Python libraries that extend the program’s core functionality beyond the standard library. These components enable advanced operations such as data visualization, spreadsheet handling, and efficient data management, all of which are essential for delivering a smooth and feature-rich user experience. While BahtBuddy is intentionally designed to minimize dependency complexity, a few carefully selected third-party modules are necessary to ensure robust performance and user convenience.

The most fundamental dependency is Tkinter, a standard GUI toolkit for Python used to construct the visual interface of BahtBuddy. Although it is typically included with Python distributions, it serves as a vital third-party component in the system’s design. Tkinter provides the framework for building interactive windows, buttons, menus, and input fields, allowing users to interact intuitively with the application without needing to use command-line operations.

***3.4 Required graphical assets, configuration and other non-program files***

N/A

***3.5 Documentation files to be provided***

The primary documentation file is the **README.md**, which serves as a quick-start guide for new users. It provides essential information such as system requirements, installation instructions, and an overview of BahtBuddy’s core features. The README also outlines common troubleshooting tips and command examples, ensuring that users can get the application running smoothly even without extensive technical knowledge. For developers, it serves as an entry point to understand the project’s structure and setup process when working directly from the source code.

The **LICENSE** file provides the legal framework governing the distribution and use of BahtBuddy. It outlines the permissions, limitations, and conditions under which the software can be used, modified, or redistributed. Including this document is important to comply with open-source standards and to protect both the rights of the developers and the freedoms of the end users.

***3.6 Development files and components that must be excluded***

**4. Additional Code Required for Deployment**

For the deployment of BahtBuddy, several additional scripts and auxiliary programs are required to streamline the packaging, installation, and initialization processes across different operating systems.

To begin with, a Windows batch script (build\_windows.bat) will be developed to automate the build process on Windows systems. This script will invoke PyInstaller to package the Python source files into a standalone executable (BahtBuddy.exe) while including all required dependencies such as Tkinter and SQLite. The batch script will also copy necessary assets ( configuration files, and database files) into the output directory and create a clean, organized folder structure for distribution. By running this script, developers can generate a complete build with a single command, significantly simplifying the deployment process.

For macOS and Linux platforms, a shell script (build\_unix.sh) will serve a similar purpose. This script will automate the build process using the corresponding PyInstaller command adapted for Unix environments. It will also set file permissions, ensure that the application’s executable is properly signed (for macOS), and place all dependencies in their respective directories. These scripts ensure platform independence and make it easy to reproduce identical builds across systems.

Additionally, an optional post\_install.py script will be included to perform initial configuration tasks after installation. This script will automatically generate the default configuration file (settings.json), initialize the SQLite database (bahtbuddy.db), and verify that the required files are properly created. This automation ensures that users can immediately start using BahtBuddy after installation without manually setting up directories or files.

**5. Deployment Tasks**

The first task in the deployment process is to verify that the development environment is correctly set up. This involves ensuring that Python 3.10 or higher is installed and that the necessary dependencies—such as Tkinter and SQLite—are functioning properly. Once the environment has been confirmed, the next step is to clone or update the project repository from GitHub to ensure that the latest stable version of the codebase is being packaged.

After verifying the source code, the development team will run a dependency check using pip install -r requirements.txt to confirm that all required Python libraries are installed and compatible. Following this, the build process begins using PyInstaller, which packages the main application files (main.py, gui.py, database.py, and validation.py) into a single standalone executable. This ensures that users will not need to manually install Python or any external packages.

Once the executable is generated, the next task is to create a clean deployment directory that contains only the compiled application, necessary configuration files (such as settings.json), and documentation files (README.md, LICENSE). All temporary build files, caches, and development artifacts are excluded to keep the package lightweight and professional.

The deployment team will then conduct local testing by running the compiled application on clean virtual environments for Windows, macOS, and Linux to ensure that the program launches correctly, saves user data, and exits without errors. After successful testing, the final build will be uploaded to GitHub Releases along with the version number, release notes, and installation instructions.

The final task is to update the project documentation—including the README and change log—to reflect the release version and any changes in functionality. This versioned release will then be distributed for download. By keeping deployment tasks simple and sequential, BahtBuddy maintains its goal of being a lightweight, easy-to-install, and maintenance-friendly desktop application for everyday financial management.

**6. Deployment Test Plan**

The deployment test plan for BahtBuddy is intentionally minimal and platform-focused. First, a clean-environment smoke test will be performed on Windows 10+, macOS 12/13, and Ubuntu 20.04+/openSUSE Leap 15 by downloading the release artifact from GitHub Releases, extracting it, and launching the bundled executable by double-clicking. The application must open without requiring Python or any additional installs, render the main window, and remain responsive for basic navigation. Immediately after launch, a first-run initialization test will verify that the program can create or locate its local SQLite database file in a writable directory and proceed without manual configuration; if the database is missing, the app should still start cleanly and remain usable.

With the app open, a core CRUD path test will validate everyday use: add a simple income transaction (credit income → debit asset) and a simple expense transaction (credit asset → debit expense), then confirm that both entries appear in the Transactions view. Following each entry, a balance update test will confirm that the relevant account balances reflect the change exactly and that totals remain consistent. A persistence test will then close and reopen the app to ensure previously saved entries reappear and balances match, demonstrating durable storage across restarts. Basic input validation tests will cover common mistakes (empty amount, non-numeric amount, malformed dates, and selecting the same account for debit and credit) to ensure the UI rejects bad input with a clear message and the app does not crash or write partial data.

A platform parity test will repeat the same create→verify→restart flow on all supported platforms and compare outcomes. The executable must launch without terminal use, transactions must save and reload identically, and balances must compute the same across Windows, macOS, and Linux. To harden deployment behavior, a read-only location test will attempt to run the executable from a non-writable directory; BahtBuddy should fail gracefully with an explanatory message directing the user to run from a writable location rather than crashing. Finally, a regression sanity test will run the same CRUD and persistence steps against the previous stable release to confirm there are no functional regressions in launch behavior, data entry, balance math, or restart persistence. Passing these tests on each platform will demonstrate that the packaged build is self-contained, robust, and aligned with BahtBuddy’s no-overengineering deployment approach.