Tax Liability Forecaster: Android Mobile Application

A project presented to the National University in partial fulfillment of the requirement for the degree of Bachelor of Science (Hon's) in Computer Science & Engineering.

Supervised By

Poly Bhoumik

Senior Lecturer, Department of

Computer Science & Engineering,

Daffodil Institute of IT (DIIT)

Submitted By

Rabeya Ahmed

Registration No: 17502004925

Session: 2017-2018

Department of Computer Science & Engineering
Daffodil Institute of IT, Dhaka
Under National University, Bangladesh

Submission date: September, 2023.

DECLARATION

I affirm that the project work titled “Tax Liability Forecaster” being submitted in partial fulfillment for the degree of B.Sc. (Hon’s) in Computer Science & Engineering is the original work carried out by me. It has not formed the part of any other project work submitted for any degree or diploma, either in this or any other University.

Submitted By

Rabeya Ahmed

Registration No: 17502004925

Session: 2017-2018

APPROVAL

The Project "Tax Liability Forecaster" is submitted to the Department of Computer Science & Engineering, DIIT under the National University of Bangladesh in partial fulfillment of the requirements for the degree of Bachelor of Science (Hon's) in Computer Science and Engineering and approved as to its style and content.

Examiner

Examiner

Poly Bhoumik
Senior Lecturer, Department of
Computer Science & Engineering,
Daffodil Institute of IT (DIIT)

Md. Imran Hossain
Head of the Department,
Computer Science & Engineering,
Daffodil Institute of IT (DIIT)

ACKNOWLEDGEMENTS

Despite our efforts, the success of this project depends largely on the encouragement and guidance of our mentors. We would like to take this opportunity to express our gratitude to the people who are playing vital role in the successful completion of this project.

Our sincere thanks to **Prof. Dr. Mohammed Shakhawat Hossain**, Principal, DIIT who has allowed us to work on this project and showed encouragement given to us.

Our cordial & heartiest thanks to our Project Supervisor **Poly Bhoumik**, Senior Lecturer, Department of Computer Science & Engineering, DIIT for her valuable guidance and support to meet the successful completion of our project and also his patronage and giving us an opportunity to undertake this Project.

We express our gratitude to **Md. Imran Hossain**, Head of Department, Computer Science & Engineering, DIIT for providing us the facilities to do the project successfully.

Our heartiest thanks to **Md. Saidur Rahman**, Co-Ordinator & Senior Lecturer, Department of Computer Science & Engineering, DIIT for his patronage and for giving us an opportunity to undertake this Project.

We express our gratitude to **Safrun Nesa Saira**, Lecturer, Department of Computer Science & Engineering, DIIT for providing us with the facilities to do the project successfully.

We express our gratitude to **Nusrhat Jahan Sarker**, Lecturer, Department of Computer Science & Engineering, DIIT for providing us with the facilities to do the project successfully.

We express our gratitude to **Md. Mizanur Rahman**, Lecturer, Department of Computer Science & Engineering, DIIT for providing us with the facilities to do the project successfully.

We express our gratitude to **Moumita Akter**, Lecturer, Department of Computer Science & Engineering, DIIT for providing us with the facilities to do the project successfully.

We express our gratitude to **Md. Mushfiqur Rahman**, Lecturer, Department of Computer Science & Engineering, DIIT for providing us with the facilities to do the project successfully.

We extend our sincere thanks to our family & classmates for their constant support throughout this project.

Finally, we would be grateful to National University, Bangladesh and coordinators of the Bachelor of Science in Computer Science and Engineering degree program for giving us this opportunity to apply the knowledge that we have gained throughout the study of degree program.

ABSTRACT

The Tax Liability Forecaster is a comprehensive digital tool designed to streamline the tax payment process for individuals and businesses. It includes a range of features such as income tax return filing for both salaried and normal returns, income tax registration, sales tax registration, online verification services, and VAT and tax calculations. Additionally, the tool offers an e-payment system that allows users to make income tax, sales tax, and federal excise duty payments with easy record-keeping capabilities. To encourage timely payments, the tool also includes a penalty calculation feature. The Tax Liability Forecaster provides a more efficient and effective way for users to manage their tax obligations, saving time and effort while improving overall tax compliance. With this project book, we aim to provide a detailed overview of the Tax Liability Forecaster, its features and benefits, and how users can use it to improve their tax payment experience.

Table of Contents

Chapter 01
Introduction

1.1 Introduction

The payment of taxes is a crucial aspect of any economy, as it provides the government with the necessary funds to support public services and infrastructure. However, navigating the complex tax payment process can be a daunting task, with many individuals and businesses finding it challenging to accurately calculate their tax liabilities and make timely payments. To address these challenges, we have developed a comprehensive digital tool called the Tax Liability Forecaster. The Tax Liability Forecaster offers a range of features to simplify the tax payment process, including income tax return filing for both salaried and normal returns, income tax registration, sales tax registration, online verification services, and the ability to calculate VAT and tax. Additionally, our tool includes an e-payment system that allows users to make income tax, sales tax, and federal excise duty payments with easy record-keeping capabilities. We understand that timely payments are critical to avoiding penalties, so our tool also includes a penalty calculation feature to help users stay on track. The Tax Liability Forecaster provides a convenient and accurate way for users to manage their tax obligations, whether they are individuals or businesses. In this project book, we will provide a detailed overview of the Tax Liability Forecaster, explaining its key features and benefits, and how users can use it to simplify their tax payment process. With the Tax Liability Forecaster, we aim to save users time and effort, improve tax compliance, and ultimately contribute to a more efficient and effective taxation system.

1.2 Project Objectives

The Tax Liability Forecaster has several objectives that guide its development and implementation. One of the primary objectives is to simplify the tax payment process for individuals and businesses, making it easier for them to calculate their tax liabilities accurately and make timely payments. . With these objectives in mind, we have identified several specific goals for the Tax Liability Forecaster project book, that including:

- To provide a detailed overview of the Tax Liability Forecaster, explaining its features and benefits, and how it can help users manage their tax obligations more effectively.
- To highlight the different features of the Tax Liability Forecaster, such as income tax return filing for both salaried and normal returns, income tax registration, sales tax registration, online verification services, and the ability to calculate VAT and tax.
- To explain the e-payment system included in the Tax Liability Forecaster, which allows users to make income tax, sales tax, and federal excise duty payments with easy record-keeping capabilities.
- To demonstrate how the Tax Liability Forecaster can help users stay on track with their tax obligations, using the penalty calculation feature.

- To provide a step-by-step guide for users on how to use the Tax Liability Forecaster effectively, whether they are individuals or businesses.
- To discuss the benefits of the Tax Liability Forecaster, such as saving time and effort, improving tax compliance, and contributing to a more efficient and effective taxation system.

1.3 Project Feature

The Tax Liability Forecaster is a powerful digital tool designed to simplify the tax payment process for individuals and businesses. It offers a wide range of features to help users accurately calculate their tax liabilities and make timely payments, while also providing additional resources to improve tax compliance. Some of the key features of the Tax Liability Forecaster include:

- Income Tax Return Filing: The Tax Liability Forecaster allows for the easy filing of both salaried and normal income tax returns, streamlining the process for users.
- Income Tax Registration: Users can quickly and easily register for income tax through the Tax Liability Forecaster, simplifying the registration process.
- Sales Tax Registration: The tool also provides sales tax registration, making it easier for businesses to comply with relevant tax laws and regulations.
- Online Verification Services: Users can verify their tax information through the Tax Liability Forecaster, ensuring that their tax payments and returns are accurate and up-to-date.
- VAT and Tax Calculation: The Tax Liability Forecaster includes a comprehensive calculator for calculating VAT and tax, taking into account relevant tax laws and regulations.
- E-Payment: The tool allows for easy and secure electronic payment of income tax, sales tax, and federal excise duty, providing users with the convenience of making payments directly through their bank accounts using Alternate Delivery Channels.
- Record-Keeping: The Tax Liability Forecaster provides users with a simple and efficient way to keep track of their tax payments, helping them stay organized and avoid penalties for late or missed payments.

1.4 Benefits

The Tax Liability Forecaster offers several benefits to individuals and businesses alike, streamlining the tax payment process and making it easier to meet tax obligations. One of the primary benefits of the Tax Liability Forecaster is its ability to accurately calculate tax liabilities, reducing the risk of errors and ensuring that individuals and businesses pay the correct amount of taxes. Additionally, the Tax Liability Forecaster's e-payment system and record-keeping capabilities make it easier

for users to make timely payments and keep track of their tax obligations. The Tax Liability Forecaster also provides valuable online verification services, simplifying the registration process for income and sales taxes. By simplifying and streamlining the tax payment process, the Tax Liability Forecaster saves users time and effort, reducing the stress associated with tax compliance.

Furthermore, by improving tax compliance, the Tax Liability Forecaster contributes to a more efficient and effective taxation system overall.

Benefits includes:

- Accurately calculates tax liabilities, reducing the risk of errors.
- Simplifies tax payment process with e-payment options and record-keeping capabilities.
- This tax Android apps are free or low-cost, which can save you money on expensive tax preparation software or hiring a tax professional
- Tax apps send reminders and notifications about important dates, ensuring you stay on top of your tax obligations. This proactive approach helps you avoid late filings and penalties.
- Saves users time and effort, reducing the stress associated with tax compliance.
- Improves tax compliance and contributes to a more efficient and effective taxation system.

1.5 Limitation of existing system

The existing tax payment system has several limitations that can make it challenging for individuals and businesses to accurately calculate their tax liabilities and make timely payments. One of the primary limitations is the complexity of the tax code, which can be difficult to navigate and understand for many taxpayers. The tax code is often subject to frequent updates and changes, which can further complicate the process of calculating tax liabilities.

Additionally, the existing tax payment system often relies on manual processes, including paper forms and in-person visits to government offices, which can be time-consuming and inconvenient for taxpayers.

Finally, the existing system often lacks transparency and accountability, which can erode trust in the taxation system and undermine compliance.

Here are the limitations of the existing system:

- Complexity of the tax code, making it difficult to navigate and understand for many taxpayers.
- Frequent updates and changes to the tax code that further complicate the process of calculating tax liabilities.
- Reliance on manual processes, including paper forms and in-person visits to government offices, that can be time-consuming and inconvenient for taxpayers.

- Lack of a centralized tax payment system, making it difficult for taxpayers to keep track of their payments and obligations.
- Lack of transparency and accountability, eroding trust in the taxation system and undermining compliance.

Chapter 02

Background Study

2.1 Background Study

The Tax Liability Forecaster is a powerful tool that can greatly simplify the tax payment process for individuals and businesses. However, to fully understand the potential benefits of this tool, it is important to first understand the broader context of tax policy and administration. Taxation is a fundamental aspect of modern societies, enabling governments to provide essential public services and infrastructure. Taxation also plays a crucial role in promoting economic stability and growth, by ensuring that governments have access to the resources they need to support social programs, public works, and investment in new initiatives. At the same time, tax policy and administration can be complex and confusing, with multiple layers of legislation and regulation that can be difficult to navigate for those who are unfamiliar with the system. This can create barriers to compliance, particularly for individuals and small businesses who may lack the resources or expertise to manage their tax obligations effectively. In addition, the consequences of non-compliance can be severe, ranging from monetary penalties to legal action and reputational damage. To address these challenges, tax authorities around the world have sought to adopt new technologies and digital tools to streamline the tax payment process and promote greater compliance. The Tax Liability Forecaster is one such tool, designed to enable individuals and businesses to calculate their tax obligations quickly and easily, and to make payments online via a range of payment options. By providing a user-friendly and efficient platform for tax compliance, the Tax Liability Forecaster can help to reduce the burden of tax administration on individuals and businesses, while also supporting governments in their efforts to ensure that everyone pays their fair share. In this chapter, we will provide a detailed background study of tax policy and administration, covering the legal frameworks and processes involved in taxation, the challenges of tax compliance, and the role of digital tools and technology in improving tax administration. This background study will provide essential context for understanding the value of the Tax Liability Forecaster, and will demonstrate how this tool can support greater efficiency, transparency, and fairness in the tax system.

Overall, the Tax Liability Forecaster represents a major step forward in the field of tax administration, providing individuals and businesses with the tools they need to manage their tax obligations effectively and with confidence.

2.2 Applicability of this System

The Tax Liability Forecaster is designed to be a user-friendly, accessible tool for individuals and businesses to accurately calculate and pay their taxes. The system is applicable to a wide range of taxpayers, including salaried employees, small business owners, and larger corporations. Additionally, the system offers a variety of features to simplify the tax payment process, including online payment options and record keeping. Applicability of our system:

- The Tax Liability Forecaster is a user-friendly, accessible tool for individuals and businesses to accurately calculate and pay their taxes.

- The system is applicable to a wide range of taxpayers, including salaried employees, small business owners, and larger corporations.
- The system can calculate income tax, sales tax, and federal excise duty, making it an all-in-one solution for tax compliance.
- The system offers a variety of features to simplify the tax payment process, including online payment options and record keeping.
- The Tax Liability Forecaster helps taxpayers avoid penalties for late or incorrect payments by calculating the correct amount of taxes owed based on current tax laws and regulations.
- The system provides alerts for upcoming payment deadlines to help prevent costly penalties and fines and ensure that taxpayers remain in good standing with tax authorities.
- The Tax Liability Forecaster offers a practical, effective solution for individuals and businesses looking to simplify the tax payment process and ensure compliance with tax regulations.
- Its user-friendly interface and comprehensive features make it an ideal choice for taxpayers of all types.

2.3 Challenges in background processing

While the Tax Liability Forecaster can greatly simplify the tax payment process for individuals and businesses, there are a number of challenges associated with background processing that must be carefully considered. One of the primary challenges is ensuring the accuracy and reliability of tax data, particularly when dealing with large volumes of information from a variety of sources.

Challenges in background processing includes:

- Background processing for tax data presents a number of challenges that must be carefully considered.
- Ensuring the accuracy and reliability of tax data is a primary challenge, particularly when dealing with large volumes of information from a variety of sources.
- Data validation and normalization are critical components of background processing, and any errors or inconsistencies can have significant consequences for tax compliance and revenue collection.
- Ensuring the security and privacy of sensitive tax data is another key challenge, particularly when it is being transmitted over digital networks or stored in cloud-based systems.
- Cyber security threats are a major concern in the modern digital landscape, and robust security measures must be in place to prevent data breaches or other unauthorized access.

2.4 Feasibility Study

Feasibility study is an important step in the software development process. It involves assessing whether a proposed project is technically, economically, and operationally feasible.

- **Technical Feasibility:** We can strongly say that it is technically feasible since there will not be much difficulty in getting the required resources for the development and maintaining the system as well.
- **Economic Feasibility:** The development of this application is highly economically feasible. The organization needed not to spend much money on the development of the system already available. we can attain the maximum usability of the corresponding resources. Even after the development, the organization will not be in a condition to invest more in the organization. Therefore, the system is economically feasible.
- **Behavioral Feasibility:** The proposed system is also behaviorally feasible as it is very user-friendly. Extensive training of the users is not required. The users can easily learn to use the system and can adapt themselves according to the system.
- **Operational Feasibility:** It is mainly related to human organizations and political aspects. The questions to be considered are:
  1. What changes will be brought with the system?
  2. What organization structures are disturbed?
  3. What new skills will be required?

  The system is operationally feasible as it is very easy for the End users to operate it. It only needs basic information about the Windows platform.
- **Schedule feasibility:** Time evaluation is the most important consideration in the development of a project. The time schedule required for the development of this project is very important since more development time affects machine time, and costs and causes delays in the development of other systems.

2.5 Summary

The background study chapter provides an in-depth overview of the Tax Liability Forecaster, a digital tool designed to simplify the tax payment process for individuals and businesses. It outlines the key objectives of the project, including improving tax compliance and making tax payment easier and more efficient for users.

Chapter 03

Literature Survey

3.1 Introduction to Literature Survey

As part of our research into the development of the Tax Liability Forecaster, we conducted a thorough literature survey to gain insights into existing tax payment systems and identify best practices. In this section of the project book, we will summarize the key findings from our literature survey, drawing on a range of sources to provide a comprehensive overview of the current state of tax payment systems and best practices for developing effective tools like the Tax Liability Forecaster.

3.2 Feature-based approach

One of the most significant advantages of the Tax Liability Forecaster is its feature-based approach, which offers users a wide range of tools and capabilities to manage their tax obligations. The e-payment feature, in particular, is a game-changer, allowing users to make income tax, sales tax, and federal excise duty payments directly through their bank accounts using alternate delivery channels. The Tax Liability Forecaster also includes a penalty calculation feature, which can help users avoid costly penalties by providing them with real-time information on their tax obligations.

The feature-based approach involves the following steps:

- **User requirements analysis**: This step involves understanding the needs and requirements of the users and identifying the key features that are essential for an effective medical sample collection from home application.
- **Feature identification**: This step involves identifying the key features that are required for the application based on the user requirements analysis. The features are prioritized based on their importance and relevance.
- **Design and development**: This step involve designing and developing the application based on the identified features. The application is developed in a user-friendly and intuitive manner, ensuring ease of use and accessibility.
- **Testing and evaluation**: This step involves testing the application to ensure that it is functioning as intended and meets the user requirements. The application is evaluated for its effectiveness, user-friendliness, and overall usability.

3.3 Authentication

In Flutter, authentication is the process of verifying the identity of a user. This is typically done by requesting the user to enter their credentials, such as a username and password, and then comparing those credentials against a pre-existing database of authorized users. Once the credentials have been validated, the user can be granted access to certain resources or features within the app. Flutter offers several built-in authentication mechanisms that can be used to implement secure login and registration flows within an app.

These include:

- **Email/Password authentication**: This mechanism uses Firebase Authentication to allow users to sign in with an email address and password that they have previously registered with the app. The user's credentials are verified against the Firebase Authentication service, which provides a secure and reliable way to store and manage user authentication data.
- **Social Sign-In authentication**: This mechanism allows users to sign in to an app using their social media accounts, such as Google, Facebook, or Twitter. This is typically done using O Auth 2.0 authentication, which allows the app to authenticate the user by requesting permission to access their social media profile information.
- **Biometric authentication**: This mechanism uses the device's biometric sensors, such as fingerprint or face recognition, to authenticate the user. This provides an additional layer of security and convenience, as users can quickly and easily authenticate themselves without having to enter a username and password.

3.4 Income Tax Return Filing (Salaried & Normal Return) system

Income tax return filing is a process through which individuals and businesses report their annual income and tax liability to the tax authorities. The process of filing an income tax return can vary depending on the nature of the income and the taxpayer's status, but generally, it involves providing detailed information about income from various sources, deductions, and credits that reduce the tax liability, and any tax payments made during the year. In the case of salaried individuals, income tax return filing is relatively straightforward. Salaried individuals receive a Form 16 from their employer, which summarizes the income earned and tax deducted at source during the year. Using this information, the salaried individual can file their income tax return by either manually filling out a physical form or electronically filing the return through the tax department's website or other authorized service providers. For individuals with income from sources other than salary, such as rental income, business income, or capital gains, filing an income tax return can be more complex.

3.5 Income Tax Registration

Income Tax Registration is a system in which an individual or business registers with the government to become a taxpayer and obtain a unique Taxpayer Identification Number (TIN). This TIN is used to identify taxpayers and track their tax payments and liabilities. In most countries, income tax registration is mandatory for anyone who earns taxable income. To register for income tax, individuals and businesses typically need to provide basic personal and financial information, such as their name, address, and source of income. Once registered, they will receive a TIN and will be required to file regular tax returns, reporting their income and any tax owed. The purpose of

income tax registration is to ensure that everyone who is required to pay taxes is properly identified and held accountable for their tax obligations. It also helps the government to collect taxes more efficiently and accurately, reducing the risk of tax evasion and ensuring that tax revenues are used to fund public services and infrastructure. Overall, income tax registration is an essential part of any tax system, helping to ensure that taxpayers are identified and held accountable for their tax obligations. By registering for income tax, individuals and businesses can ensure that they are in compliance with tax laws and avoid potential penalties and legal consequences for non-compliance.

3. 6 Summary

Here in this chapter, we have discussed the basic features of this project. These are the building blocks of this project. Every feature that is used in our project is described here.

Chapter 04

Methodology

4.1 Introduction to Analysis Model

Analysis model operates as a link between the 'system description' and the 'design model'. In the analysis model, information, functions, and the behavior of the system are defined, and these are translated into the architecture, interface, and component-level design in the 'design modeling'.

4.2 Software Development Life Cycle (SDLC)

Software Development life cycle (SDLC) is a spiritual model used in project management that defines the stages included in an information system development project, from an initial feasibility study to the maintenance of the completed application. There are different software development life cycle models specified and designed, which are followed during the software development phase. These models are also called "Software Development Process Models**."** Each process model follows a series of phases unique to its type to ensure success in the step of software development. [6]

Here, are some important phases of SDLC life cycle:

- Waterfall model
- Iterative model
- Spiral model
- V-shaped model
- Incremental model
- Agile model

4.3 Incremental Model

Incremental process model is also known as the Successive version model. The incremental model is a process of software development where requirements are broken down into multiple standalone modules of the software development cycle. Incremental development is done in steps from analysis, design, implementation, testing/verification, and maintenance. Each iteration passes through the requirements, design, coding, and testing phases. [7]

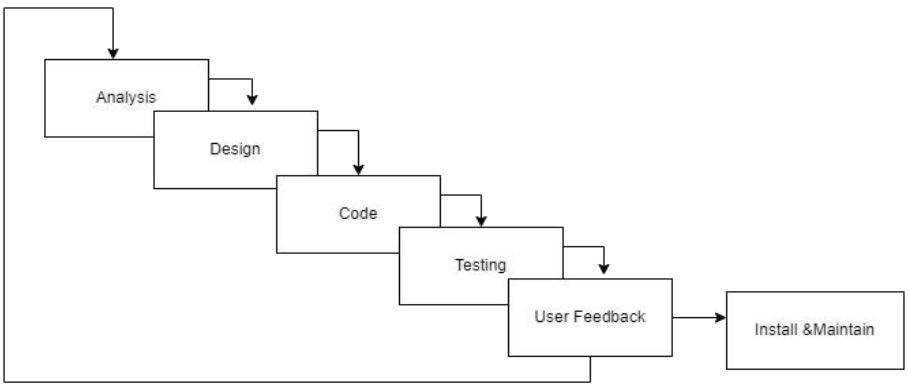


Fig 4.3: Incremental Model

4.4 Characteristics of an Incremental module

- System development is broken down into many mini development projects;
- Partial systems are successively built to produce a final total system.
- Highest priority requirement is tackled first.
- Once the requirement is developed, requirements for that increment are frozen.

4.5 When to use this

- Funding Schedule, Risk, Program Complexity, or need for early realization.
- When Requirements are known up-front.
- When projects have lengthy development schedules.
- Projects with new Technology.

4.6 Advantages

- Error Reduction (core modules are used by the customer from the beginning of the phase and then these are tested thoroughly)
- Uses divide and conquer for a breakdown of tasks.
- Lowers initial delivery cost.

4.7 Disadvantages

- Requires good planning and design.
- The total cost is not lower.

4.8 Summary

Here in this chapter, we have discussed the SDLC model that is being used to develop our Academic Research Publishing Management System project. Here we have used the Incremental model.

Chapter: 05

Requirements

5.1 Introduction to Requirements

A System Requirements Specification (SRS) is a comprehensive document that outlines the requirements of a software or system development project. It serves as a roadmap for the entire project and defines the scope, functionality, and specifications of the system or software being developed. An SRS typically includes a detailed description of the system, its features, and its intended users, as well as the hardware and software requirements needed to support the system. The primary purpose of an SRS is to ensure that all stakeholders have a clear understanding of the project's objectives and requirements before development begins. This includes developers, project managers, clients, and end-users. By defining the requirements upfront, an SRS helps to ensure that the final product will meet the needs of all stakeholders and that the project is completed on time and within budget. A typical SRS document includes several key sections, such as an introduction, scope, functional requirements, non-functional requirements, system architecture, user interface design, and testing and validation procedures.

5.1.1 Hardware Requirements

- Intel Core i5 or higher
- 8GB RAM or more

5.1.2 Software Requirements

- Android Studio
- Emulator
- Web Browser (Google chrome / Brave / Firefox)

5.2 Dart

Dart is an object-oriented, client-side programming language that was developed by Google in 2011 as a new programming language for building web and mobile applications. It is designed to be fast, scalable, and easy to use, making it a popular choice for developers who want to create high-performance, cross-platform applications. One of the main features of Dart is its use of a just-in-time (JIT) compiler. [16]

5.3 Flutter

Flutter is an open-source mobile application development framework developed by Google. It is a powerful and flexible tool for building high-performance, visually appealing, and cross-platform mobile applications for both iOS and Android platforms. Flutter uses the Dart programming language, which is also developed by

Google, and it offers a unique approach to mobile app development that allows developers to build complex and beautiful mobile applications in a shorter time frame. One of the main advantages of Flutter is its ability to offer a rich and responsive user interface that can adapt to different screen sizes, resolutions, and orientations. Flutter allows developers to create custom widgets and animations, which are rendered natively on the device, resulting in a smooth and fluid user experience. [17]

5.4 Emulator

An emulator is a software program that enables a computer system to simulate the behavior of another computer system. In the context of mobile application development, an emulator allows developers to test and run their applications on a virtual mobile device, without the need for a physical device. Emulators provide a cost-effective and efficient way to test mobile applications on different platforms and devices, without the need for purchasing multiple devices. Emulators can simulate various device configurations, such as screen size, resolution, and operating system version, enabling developers to test their applications under different conditions. The use of emulators is particularly beneficial for mobile application developers who are working on cross-platform projects or who need to test their applications on multiple devices. Emulators can help developers to reduce the time and cost involved in mobile application development and improve the overall quality of their applications. [3]

5.5 Android Studio

Android Studio is an integrated development environment (IDE) created by Google, specifically designed for Android application development. It provides developers with a comprehensive suite of tools for designing, building, testing, and deploying high-quality mobile applications for Android devices. Android Studio offers a user-friendly interface and a range of features that streamline the development process, including code completion, syntax highlighting, and debugging tools. It also offers support for multiple programming languages, including Java, Kotlin, and C++, allowing developers to choose the language that best suits their needs. One of the key advantages of Android Studio is its integration with other Google services and platforms, such as the Google Play Store, Firebase, and Google Cloud Platform. This integration enables developers to access a range of services and resources, such as analytics, crash reporting, and cloud storage, that can help them to build and scale their applications. Overall, Android Studio is a powerful and versatile tool for Android application development, providing developers with the tools and resources they need to create high-quality mobile applications for Android devices. [9]

5.6 Web Browser

A web browser is a software program that allows a user to locate, access, and display web pages. In common usage, a web browser is usually shortened to "browser." Web browsers are used primarily for displaying and accessing websites on the internet, as well as other content created using languages such as Hypertext Markup Language (HTML) and Extensible Markup Language (XML). Browsers translate web pages and websites delivered using Hypertext Transfer Protocol (HTTP) into human-readable content. They also have the ability to display other protocols and prefixes, such as secure HTTP (HTTPS), File Transfer Protocol (FTP), email handling, and files. [10]

5.7 Summary

Here in this chapter, we have discussed the hardware and software that are needed to build our project. It includes programming language IDE, framework, editor, etc.

Chapter 06

Design of Proposed System

6.1 Introduction to system overview

A system overview is a high-level description of the major components, functions, and processes that make up a system. It provides an overall understanding of how a system works and what it is designed to accomplish. In the context of software development, a system overview usually includes a description of the software architecture, user interfaces, data flows, and other key elements that contribute to the system's functionality. It can help developers and designers understand the relationships between different components of the system, identifying potential areas of weakness or inefficiency. Finally, a system overview can serve as a roadmap for future development efforts, providing a reference point for developers as they work to enhance or expand the system's capabilities. Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of the software development cycle. Incremental development is done in steps from analysis, design, implementation, testing/verification, and maintenance. Each iteration passes through the requirements, design, coding, and testing phases and each subsequent release of the system adds function to the previous release until all designed functionality has been implemented . [6]

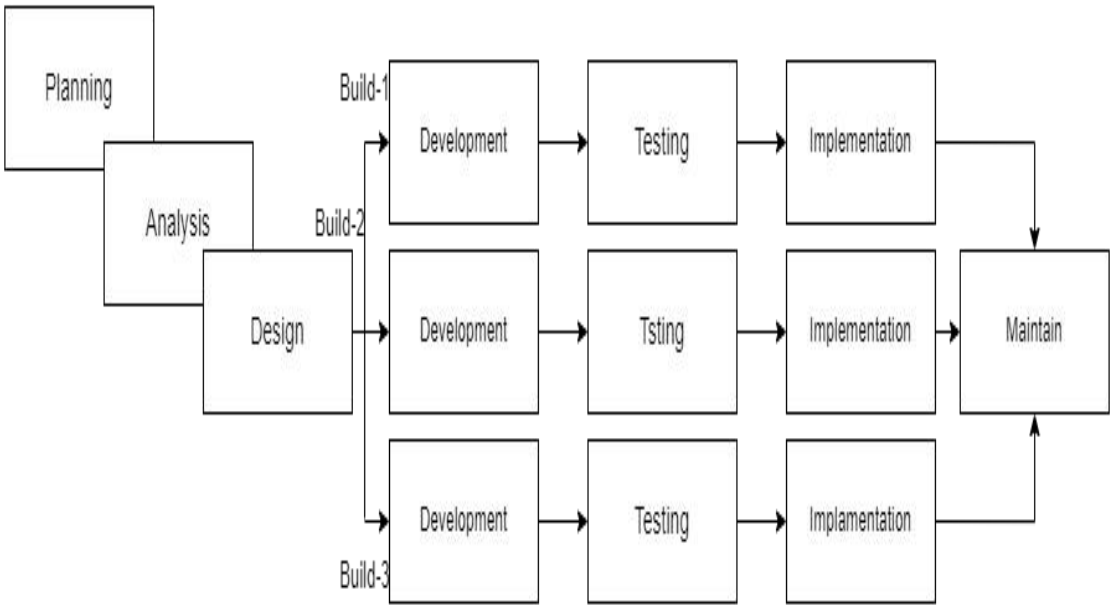


Fig 6.1: Incremental Model of SDLC

6.2 Flow Chart

An App flowchart (also known as a sitemap) maps out the structure and complexity of any current or future website. A well-structured sitemap or flowchart makes your website easily searchable. Each piece of content should ideally give users accurate search results, based on keywords connected to your web content. [14]

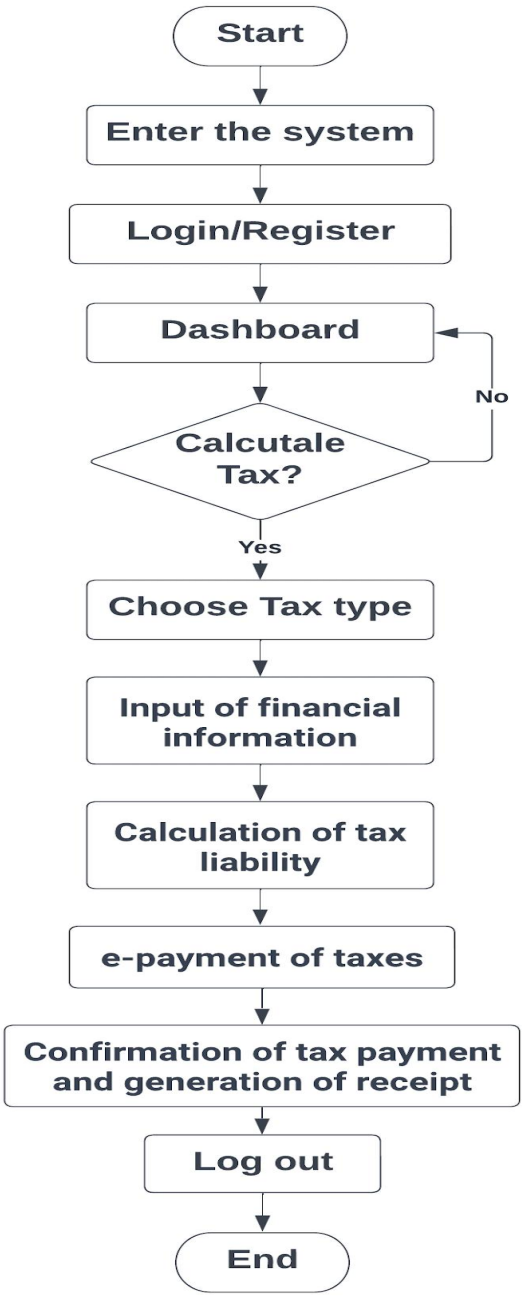


Fig 6.2 Flow Chart of Tax Liability Forecaster.

6.3 Workflow of Tax Liability Forecaster

Flow diagram is a collective term for a diagram representing a flow or set of dynamic relationships in a system. The term flow diagram is also used as a synonym for flowchart and sometimes as a counterpart of the flowchart. Flow diagrams are used to structure and order a complex system or to reveal the underlying structure of the elements and their interaction. [18]

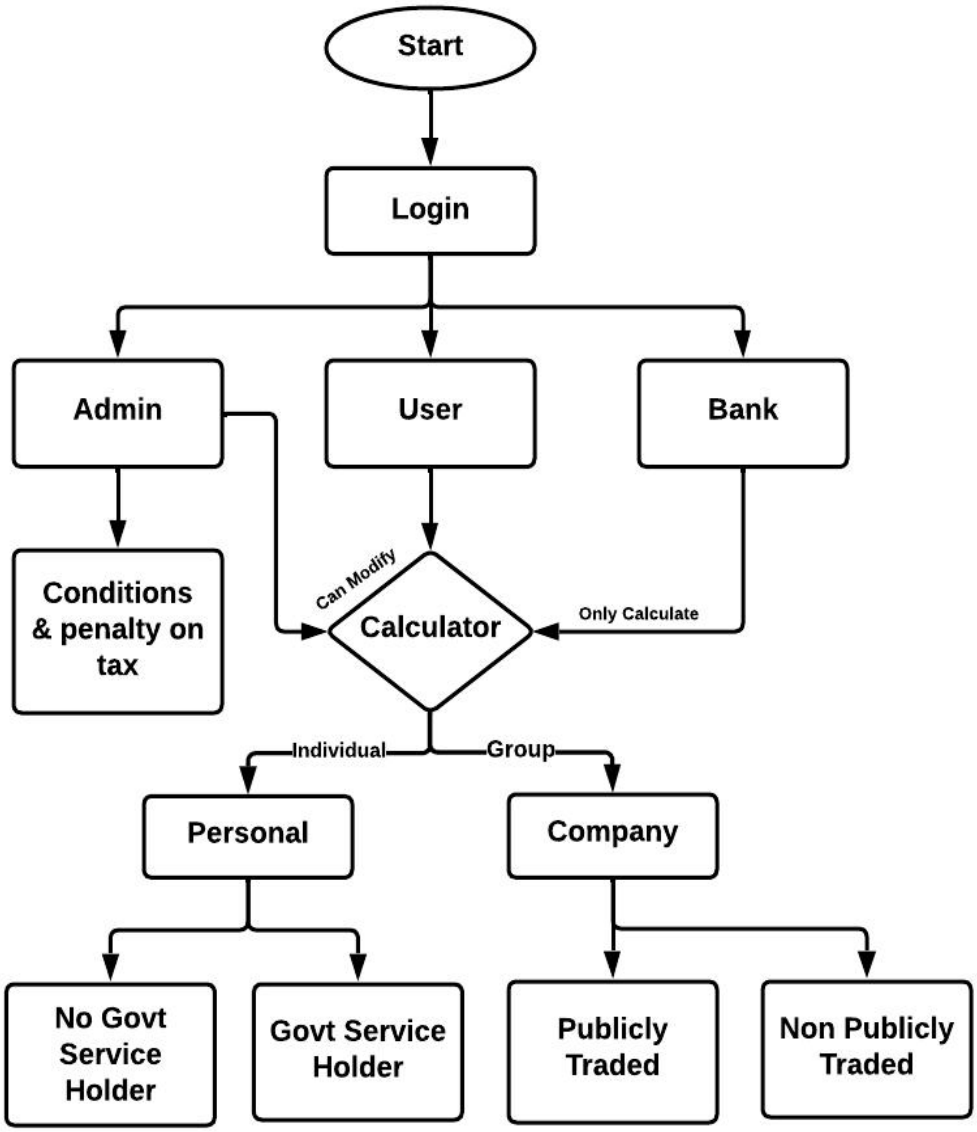


Figure 6.3: Workflow of Tax Liability Forecaster

6.4 Entity Relationship Diagram (ERD)

An Entity-Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects, or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education, and research. Also known as ERDs or ER Models. The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:

- **One to one_**– When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one.
- **Many to one_**– When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one.
- **Many to many** – When entities in all entity sets can take part more than once in the relationship, cardinality is many to many. [8]

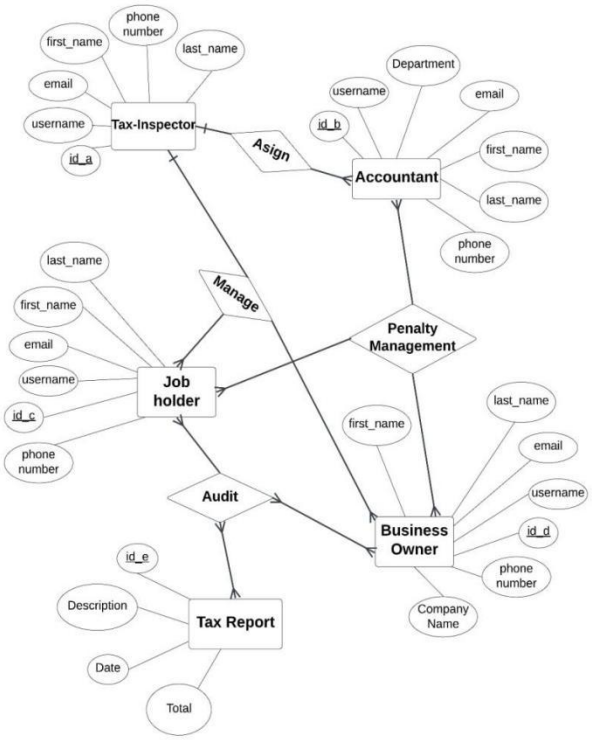


Figure 6.4: ER Diagram of Tax Liability Forecaster

6.5 Use CASE Diagram

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

Scenarios in which your system or application interacts with people, organizations, or external systems.

Goals that your system or application helps those entities (known as actors) achieve.

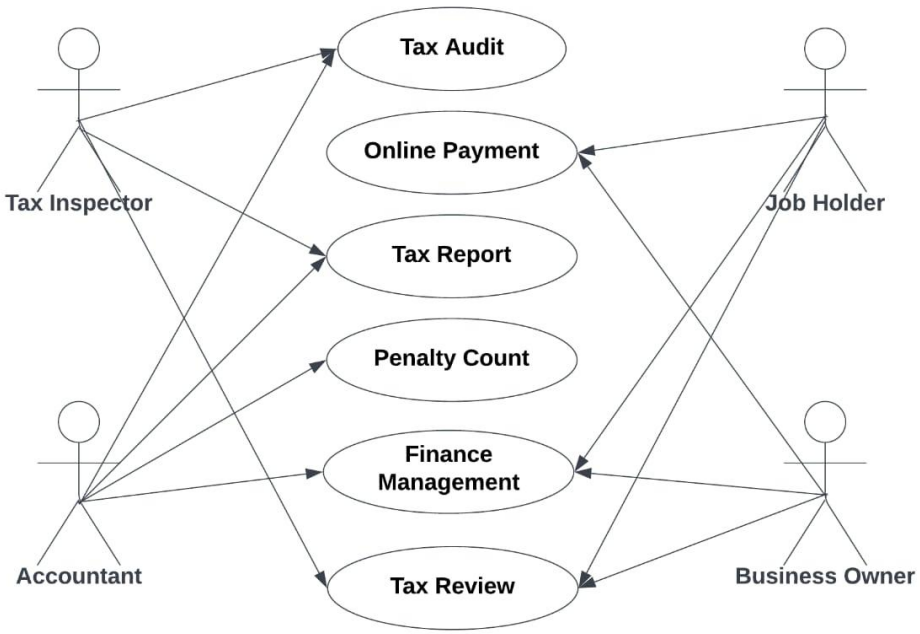


Fig 6.5: Use Case Diagram of Tax Liability Forecaster

6.6 Activity Diagram

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination.

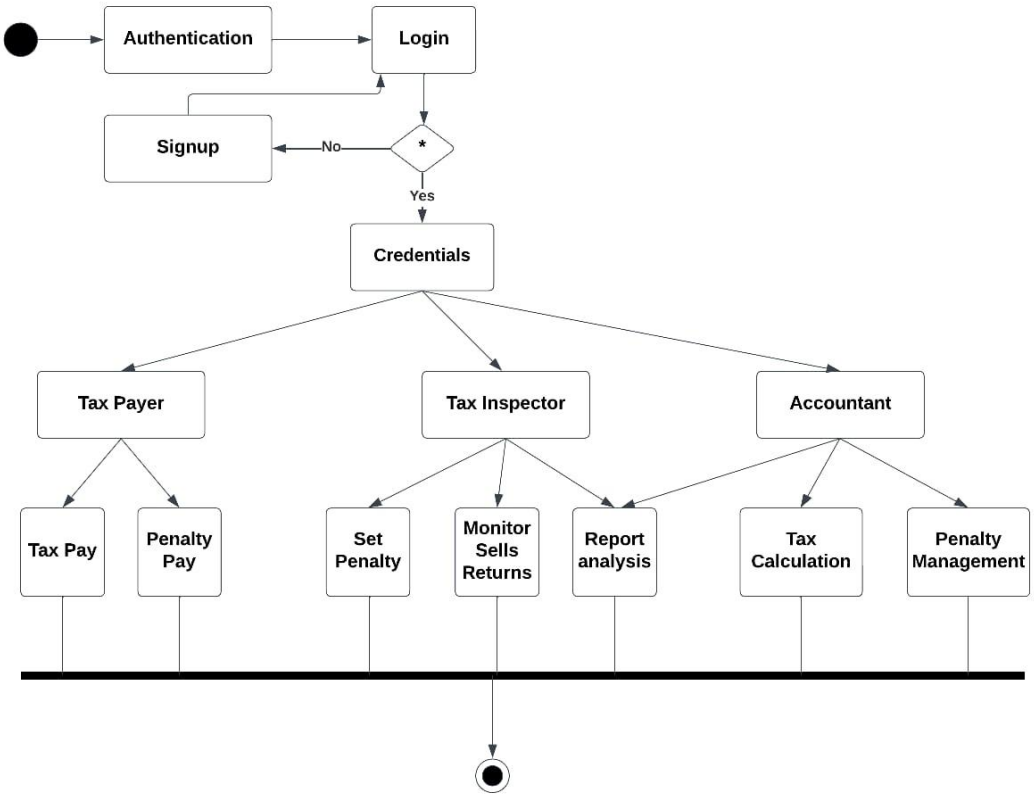


Fig 6.6: Activity Diagram of Tax Liability Forecaster

6.7 Sequence Diagram:

A sequence diagram or system sequence diagram (SSD) shows object interactions arranged in time sequence in the field of software engineering Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. For a particular scenario of a use case, the diagrams show the events that external actors generate, their order, and possible inter-system events. All systems are treated as a black box; the diagram places emphasis on events that cross the system boundary from actors to systems. A system sequence diagram should be done for the main success scenario of the use case, and frequent or complex alternative scenarios.

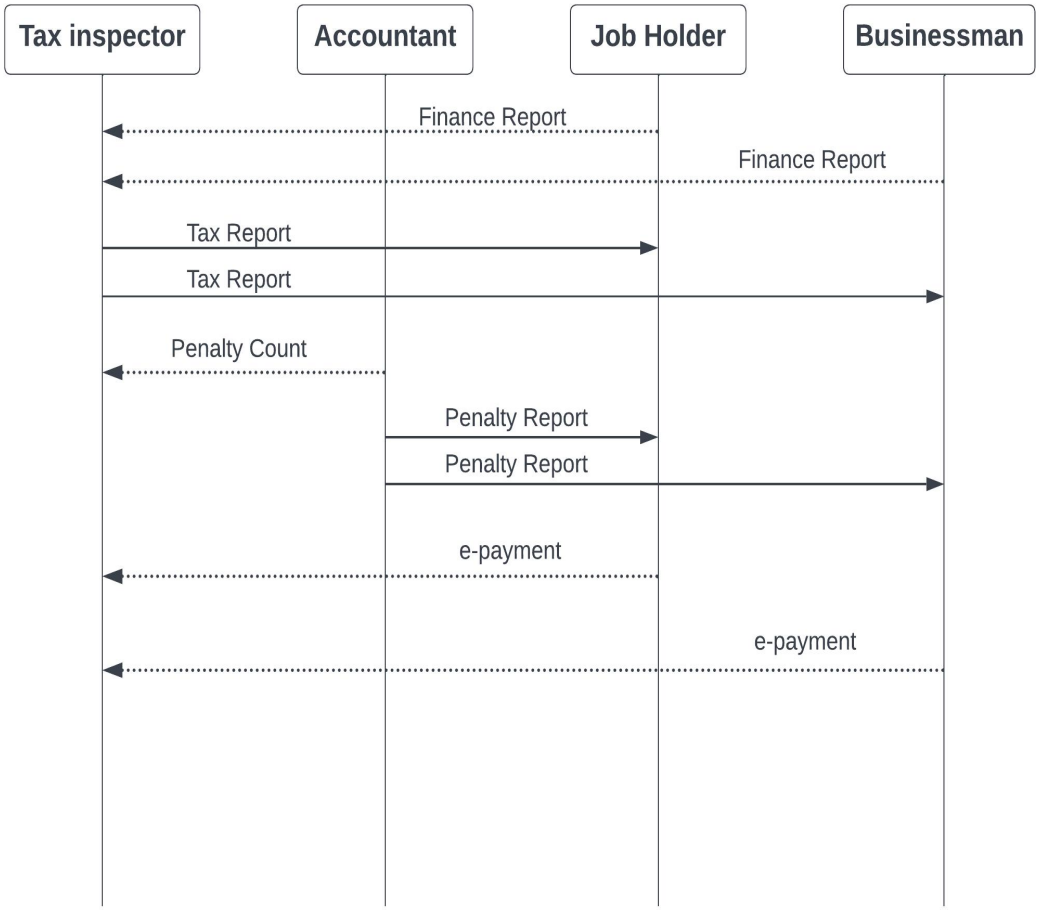


Fig 6.7: Sequence diagram of Tax Liability Forecaster

6.8 Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart). [15]

6.8.1 DFD Level 0:

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

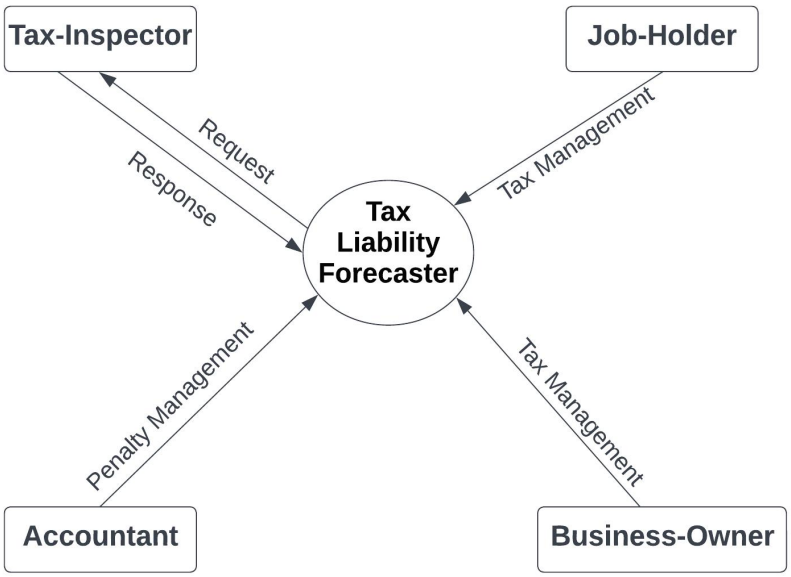


Fig 6.8.1: DFD level-0 of Tax Liability Forecaster

6.8.2 DFD Level 1:

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and break down the high-level process of 0-level DFD into sub processes.

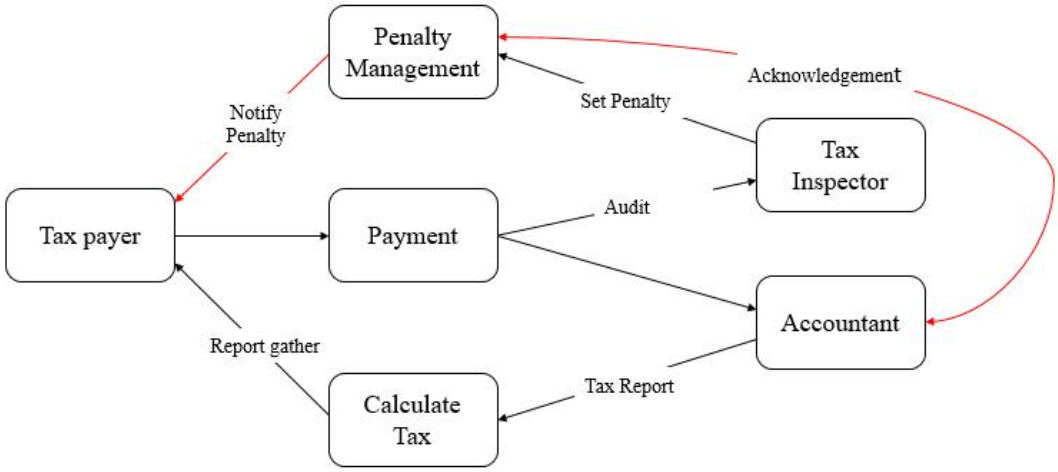


Fig 6.8.2 DFD Level 2.1 of Tax Liability Forecaster

6.8.3 DFD Level 2:

2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system's functioning.

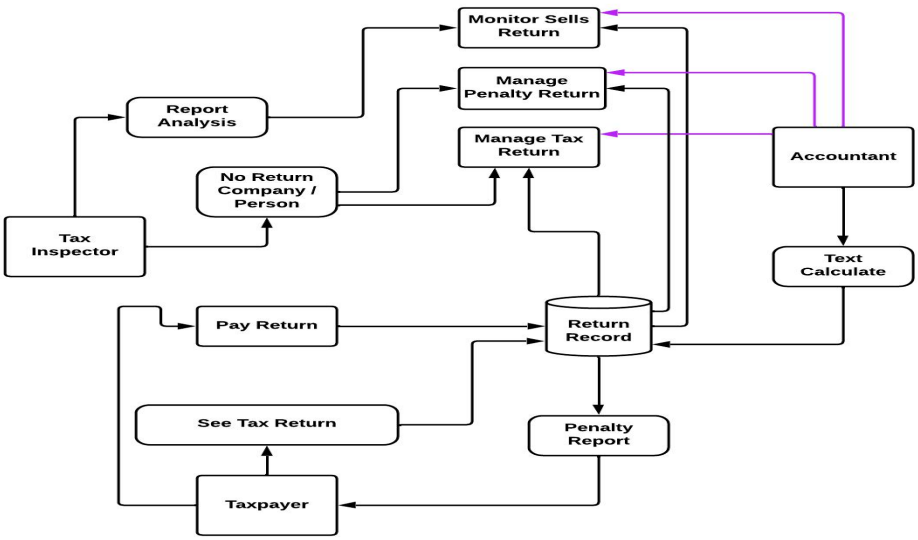


Fig 6.8.3 DFD Level 3: of Tax Liability Forecaster.

6.9 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object-oriented modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed. In the diagram, classes are represented with boxes that contain three compartments:

- The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle compartment contains the attributes of the class. They are left-aligned, and the first letter is lowercase.
- The bottom compartment contains the operations the class can execute.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. In detailed modeling, the classes of the conceptual design are often split into subclasses.

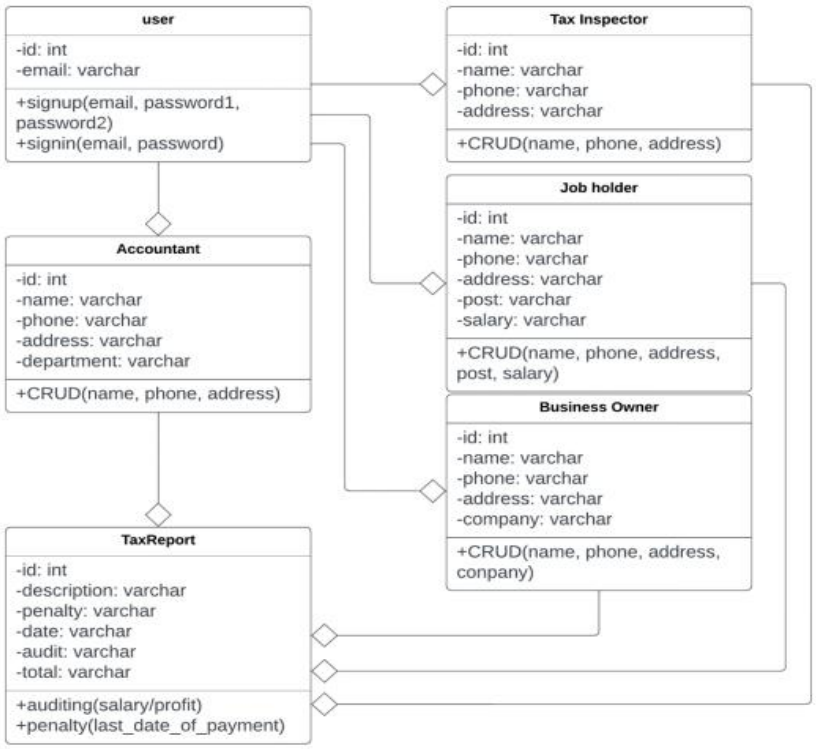


Fig 6.9: Class diagram of Tax Liability Forecaster

Chapter 7

Implementation

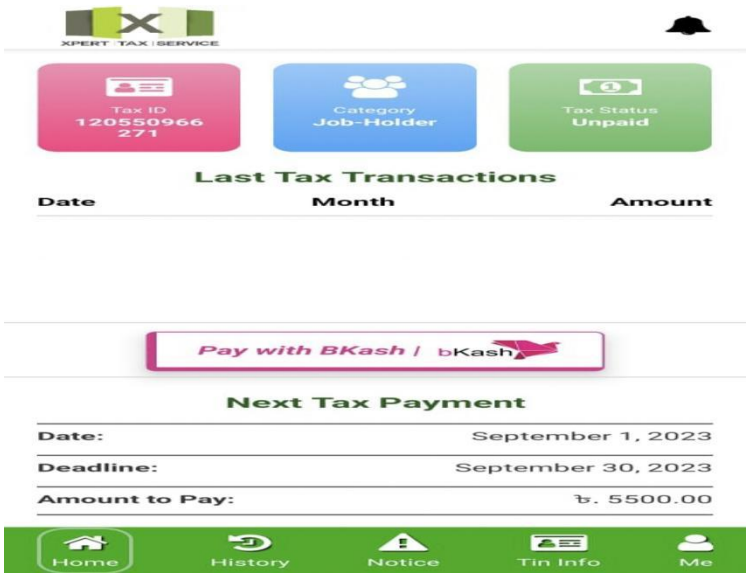
7.1 Our System Preview

Tax payer Sign Up option

7.2 Tax payer login

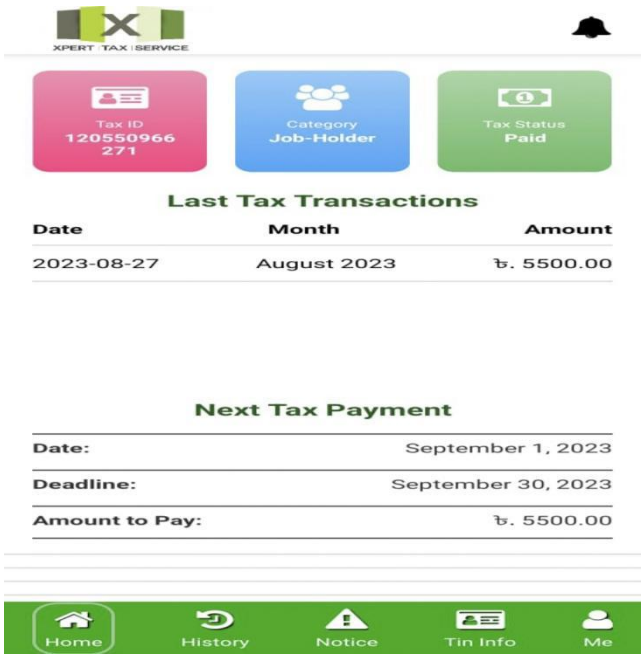
Tax payer Login option

7.3 Unpaid status Home Page



Unpaid status Home Page

7.4 Paid status Home Page



Paid status Home Page

Chapter 8

Conclusion

8.1 Business Perspective

From a business perspective, the Tax Liability Forecaster represents a significant opportunity to streamline and simplify tax payment processes, while improving compliance and minimizing the risk of penalties. For businesses, managing tax liabilities can be a complex and time-consuming task, with many companies struggling to keep up with the latest regulations and requirements. The Tax Liability Forecaster offers a range of features to address these challenges, from income tax return filing to sales tax registration and e-payment options. By using the Tax Liability Forecaster, businesses can save time, reduce administrative costs, and avoid the risk of penalties and fines for late or inaccurate payments.

Business Perspective includes:

- The Tax Liability Forecaster streamlines tax payment processes and reduces administrative costs for businesses.
- The tool helps businesses avoid the risk of penalties and fines for late or inaccurate tax payments.
- The Tax Liability Forecaster provides clear and detailed views of a company's tax liabilities and payments, enhancing financial reporting capabilities.
- The tool helps businesses manage complex tax obligations, especially those operating in multiple tax jurisdictions.

8.2 Conclusion

In conclusion, the Tax Liability Forecaster tool would depend on the accuracy and reliability of the tools calculations and predictions. We introduced the Tax Liability Forecaster, a digital tool designed to simplify the tax payment process for individuals and businesses. Tax Liability Forecaster, demonstrating how it can simplify the tax payment process, highlighting the importance of timely tax payments, providing step-by-step instructions for using the tool, and emphasizing the value of the tool in improving tax compliance.

References:

[1]. E-commerce in the time of covid-19. OECD. (2020, October 7). Retrieved April 16, 2022

[2]. Wikimedia Foundation. (2022, March 14). Authentication. Wikipedia. Retrieved April 14, 2022

[3]. "The Emulation Imitation". Malwarebytes Labs. 17 October 2014. Retrieved 2016-05-30.

[4]. Miva. (n.d.). The history of eCommerce: How did it all begin? Miva Blog - Browse Miva's Blog for expert eCommerce strategy, visual content, and pro tips for omnichannel enterprise sales. Resources and best practices for online business. Retrieved April 16, 2022

[5]. Fernando, J. (2022, February 8). Payment gateway. Investopedia. Retrieved April 14, 2022

[6].SDLC - Quick Guide. (n.d.). Retrieved April 13, 2022

[7]. Incremental model (software engineering) - javatpoint. www.javatpoint.com. (n.d.). Retrieved April 14, 2022

[8]. Wikimedia Foundation. (2022, April 12). Python (programming language). Wikipedia. Retrieved April 13, 2022

[9]. Honig, Zach (May 15, 2013). "Google intros Android Studio, an IDE for building apps". Engadget. AOL. Retrieved May 16, 2013.

[10]. What is a Browser?. Google (on YouTube). 30 April 2009. Archived from the original on 11 December 2021.

[11]. Django. Full Stack Python. (n.d.). Retrieved April 13, 2022

[13]. Forms have never been this crispy. Django. (n.d.). Retrieved April 14, 2022

[14]. Wikimedia Foundation. (2022, April 6). Flowchart. Wikipedia. Retrieved April 15, 2022

[15]. Levels in data flow diagrams (DFD). GeeksforGeeks. (2020, October 28). Retrieved April 14, 2022

[16]. https://en.wikipedia.org/wiki/Dart_(programming_language).

[17]. _"Google announces Flutter 1.0, the first stable release of its cross-platform mobile development toolkit". Android Police. 2018-12-05. Retrieved 2022-03-17.

[18]. Robert L. Harris. Information Graphics: A Comprehensive Illustrated Reference. 2000.

APPENDIX

- SDLC – Software Development Life Cycle
- ERD – Entity Relationship Diagram
- DB – Database
- HTML – Hypertext Markup Language
- UML – Unified Modeling Language
- SSD – System Sequence Diagram
- DFD – Data Flow Diagram
- IDE – Integrated Development Environment
- XML – Extensible Markup Language
- HTTP – Hypertext Transfer Protocol
- XAMPP – Extremely Accelerated Multiprocessing
- FTP – File Transfer Protocol

APPENDIX 2

Home page code

```
import React, { useEffect, useState } from 'react';
import { useFocusEffect } from '@react-navigation/native';
import { View, Text, StyleSheet } from 'react-native';
import NavBar from './NavBar';
import TopBar from './TopBar';
import TaxInformation from './TaxInformation';
import TaxTransactionTable from './TaxTransactionTable';
import PaymentButton from './PaymentButton';
import NextTaxInfo from './NextTaxInfo';
import { useCookies } from 'react-cookie';
import API_BASE_URL from '../../apiConfig';
import { useNavigation } from '@react-navigation/native';

const Home = () => {
   const [activeButton, setActiveButton] = useState('home');
   const [reloadTrigger, setReloadTrigger] = useState(0);

   const [token] = useCookies(['myToken']);
   const navigation = useNavigation();

   const [taxID, setTaxID] = useState(0);
   const [category, setCategory] = useState('');
   const [taxStatus, setTaxStatus] = useState(false);
   const [transactions, setTransactions] = useState([]);
   const [tax, setTax] = useState('');

   useEffect(() => {
     fetchData();
   }, []);


   useFocusEffect(
     React.useCallback(() => {
       fetchData();
       return () => {
       };
     }, [])
   );
```

```
  const fetchData = async () => {
   try {
     const response = await fetch(`${API_BASE_URL}/notification/user-home-data/`, {
       method: 'GET',
       headers: {
         Accept: 'application/json',
         'Content-Type': 'application/json',
         Authorization: `Bearer ${token.access_token}`,
       },
     });
     const data = await response.json();
   if (data !== null) {
       setTaxID(data.tax_id);
       setCategory(data.category);
       setTaxStatus(data.is_paid);
       setTransactions(data.transactions);
       setTax(data.tax);
     }
    } catch (error) {
     console.log('Error fetching data:', error);
    }
  };
  const handleHomePress = () => {
     setActiveButton('home');
    };

const handleReload = () => {

  setReloadTrigger(reloadTrigger + 1);
};
return (
  <View style={styles.container}>
  <TopBar />
    <View style={styles.content}>
    <TaxInformation
      taxId={taxID}
      taxCategory={category}
      taxStatus={taxStatus}
    />
```

```
      <TaxTransactionTable header={'Last Tax Transactions'}
transactions={transactions}/>
      {taxStatus === false && (
       <>
         <View style={styles.divider} />
         <PaymentButton onPress={() => {navigation.navigate('Payment', { amount:
tax });} }/>
         <View style={styles.divider} />
       </>
      )}
       <NextTaxInfo amountToPay={tax} />
      <View style={styles.divider} />
      <View style={styles.divider} />
      <View style={styles.divider} />
      <View style={styles.divider} />
    </View>
    <NavBar onHomePress={handleHomePress} activeButton={activeButton} />
  </View>
 );
};
const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: 'white',
  alignItems: 'center',
  justifyContent: 'center',
 },
 content: {
  flex: 1,
  width: '100%',
  alignItems: 'center',
  marginTop: 20,
  marginBottom: 60,
 },
 divider: {
  width: '100%',
  height: 1,
  backgroundColor: '#ccc',
  marginVertical: 7,
 },
```

```
  notificationText: {
    fontSize: 12,
    color: 'red',
    marginTop: 5,
    marginBottom: 5,
  },
});
export default Home;
```

Top Bar code

```
import React, { useEffect, useState } from 'react';
import { useFocusEffect } from '@react-navigation/native';
import { useCookies } from 'react-cookie';
import { View, Text, StyleSheet, TouchableOpacity, Image } from 'react-native';
import Icon from 'react-native-vector-icons/FontAwesome';
import { useNavigation } from '@react-navigation/native';

const TopBar = () => {

    const [token] = useCookies(['myToken']);

    const navigation = useNavigation();

    const [notifications, setNotifications] = useState(null);
    useEffect(() => {
      fetchData();
    }, []);
    useFocusEffect(
      React.useCallback(() => {
        fetchData();
        return () => {
        };
      }, [])
    );
    const fetchData = async () => {
      try {
        const response = await fetch('http://192.168.0.107:8000/notification/notification-
data/', {
          method: 'GET',
          headers: {
```

```
          Accept: 'application/json',
          'Content-Type': 'application/json',
          Authorization: `Bearer ${token.access_token}`,
        },
      });
      const data = await response.json();
      if (data !== null) {
        setNotifications(data.notifications);
      }
    } catch (error) {
      console.log('Error fetching data:', error);
    }
  };

  return (
    <View style={styles.container}>
      <View style={styles.logoContainer}>
        <Image source={require('../../../assets/logo2.png')} style={styles.logo} />
      </View>

      <TouchableOpacity style={styles.iconContainer} onPress={() =>
{navigation.navigate('Notifications');}}>
        <Icon name="bell" size={24} color="#000" />
        {notifications > 0 && (
          <View style={styles.badge}>
            <Text style={styles.badgeText}>{notifications}</Text>
          </View>
        )}
      </TouchableOpacity>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    top: 30,
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
    backgroundColor: '#fff',
```

```
    height: 60,
    paddingHorizontal: 16,
    elevation: 5,
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.1,
    shadowRadius: 2,
  },
  logoContainer: {
    flex: 1,
    alignItems: 'flex-start',
  },
  logo: {
    width: 100,
    height: 60,
    resizeMode: 'contain',
  },
  iconContainer: {
    alignItems: 'flex-end',
  },
  badge: {
    backgroundColor: 'red', // Customize the badge background color
    position: 'absolute',
    top: -5,
    right: -5,
    minWidth: 16,
    height: 16,
    borderRadius: 10,
    alignItems: 'center',
    justifyContent: 'center',

  },
  badgeText: {
    color: 'white', // Customize the badge text color
    fontSize: 12,
    fontWeight: 'bold',
    paddingBottom: 3,
  },
});
```