Heaven's Light is Our Guide

Industrial Attachment Report

By

Syed Mahmudul Imran

(2010058)

Industrial Training (ECE-4100)

Duration of Training: March 3, 2025 to March 14, 2025

Instructor:

Prof. Dr. Md. Anwar Hossain,

Head of Dept., ECE, RUET

Rajshahi University of Engineering & Technology

Faculty of Electrical & Computer Engineering

Department of Electrical & Computer Engineering

Rajshahi-6204

29 October 2025

Certificate

Acknowledgment

I am sincerely grateful to everyone who made this industrial attachment possible. My heartfelt thanks go to the management of **BJIT Ltd.** for opening their doors to us and offering a real glimpse into professional software development. I am especially thankful to our mentors and the training team — including Mr. Homayun Kabir (Head, BJIT Academy) — for their patient guidance, timely feedback, and constant encouragement throughout the program.

I would also like to acknowledge the **Department of Electrical & Computer Engineering (ECE), RUET** for integrating industrial training into the curriculum. This experience meaningfully connected classroom theory with day-to-day engineering practice and helped me understand how ideas become working systems.

My sincere appreciation goes to my academic supervisor, **Prof. Dr. Md. Anwar Hossain**, Department of ECE, RUET, for his steady guidance and support during my studies and in preparing this report.

Finally, I am deeply thankful to my family and friends for their unwavering support and motivation. The attachment has been truly enriching, and I am grateful for the skills, discipline, and perspective I have gained.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background of Industrial Training Program

Industrial training is an essential component of the undergraduate engineering curriculum at Rajshahi University of Engineering & Technology (RUET). The ECE-4100 course is intended to provide we with an opportunity to gain hands-on exposure in a professional engineering environment. This exposure is critical for grasp the real-world applications of the theoretical concepts learned in the classroom. The program aims to immerse we in the day-to-day operations of an organization, allowing them to observe and participate in actual projects, understand industry standards, and develop professional skills that are vital for a successful career in engineering. This report documents the exposure and learning acquired during a two-week industrial attachment at BJIT Ltd., a leading global software company.

During the internship, we were immersed in a professional software development setting, learning about the lifecycle of projects, from requirements gathering to implementation and testing. The background preparation for the program included basic training in Python programming (as Python was the primary language used during the attachment) and familiarization with fundamental software engineering concepts. This made sure that we could effectively participate in tasks and comprehend discussions on industry practices. The program was also aligned with ongoing academic coursework – for example, concepts from courses like software engineering, programming, and project management were revisited and applied in a hands-on context. Overall, the industrial training program's background is rooted in the need to complement academic knowledge with experiential learning. Universities and industries collaborated to design a curriculum that balances theoretical sessions (lectures, seminars by industry experts) with hands-on assignments and projects. At BJIT, the program leveraged the company's ongoing emphasis on quality and continuous improvement, reflecting the "Japanese quality" standard the company upholds [1]. This environment offered an ideal background for trainees to learn professional work ethics, high coding standards, and modern development methodologies.

1.2 Objectives of the Training

The main objectives of this industrial training were multifaceted, aiming to provide a comprehensive learning exposure. The main goals were:

- **Apply Theoretical Knowledge in Practice:** Allow students to apply programming and engineering concepts learned in classrooms (data structures, algorithms, software design, etc.) to real-world problems. For instance, using Python to solve algorithmic tasks and to develop a functional software project.

- **Gain Exposure to Software Development Life Cycle (SDLC):** Introduce students to industry-standard software development processes. This included understanding requirement analysis, writing Software Requirements Specifications (SRS), designing system architecture, coding, version control, testing, and deployment practices. During the training, we prepared an SRS and a design document for the main project, mirroring professional SDLC stages.

- **Develop Technical Skills:** Improve proficiency in specific technical areas. In this attachment, a strong focus was on Python programming (syntax, data structures, file handling, OOP, etc.), as well as tools like Git/GitHub for version control, JSON for data storage, and possibly basics of web or database if needed for the project. The training also aimed at familiarizing us with development environments and frameworks used at BJIT.

- **Learn Modern Software Tools and Collaboration:** A key objective was to have students learn to use modern collaboration tools and workflows. We were introduced to GitHub for code collaboration and learned how teams coordinate using version control. We also practiced agile project management fundamentals (Scrum framework), including working in sprints and daily stand-up meetings. This was meant to teach how to structure and manage work through iterative development and teamwork (Scrum is an agile framework that helps teams organize and manage work via a set of values, principles, and practices [2]).

- **Enhance Soft Skills and Professionalism:** The program included objectives to improve communication, teamwork, and professional etiquette. Through group assignments, scrum team formation, and direct interaction with industry professionals, we were to develop teamwork capabilities and learn workplace ethics. Sessions on soft skills (e.g., effective communication, working in high-performing teams, workplace ethics) were embedded to meet this objective.

- **Career Orientation and Networking:** Another objective was to give trainees a glimpse into corporate culture and various career paths in the IT industry. By interacting with professionals from different departments (development, quality assurance, R&D), we

could start building a professional network and gain insights that could guide our future career choices. The final day's networking session with different development teams and Q&A was particularly aimed at this.

- **Bridge Academic Knowledge with Industry Practice:** To bridge the gap between theoretical knowledge acquired in academia and its practical implementation in the industry.

By the end of the training, the expectation was that each student would have completed a project or set of assignments demonstrating their ability to learn and work on new tasks. Moreover, we should understand how an IT company operates, what expectations the industry has for fresh engineers, and how to adapt to those expectations. Overall, the objectives spanned both technical competencies and professional development, ensuring a holistic growth through the industrial attachment.

1.3 Relevance to Course and Field of Study

As a student of Electrical and Computer Engineering (ECE), this training at a software development firm is very relevant. The ECE curriculum provides a strong foundation in programming, algorithms, data structures, and system design. This industrial attachment offered a platform to apply these foundational concepts to build functional applications. The focus on software development methodologies, project management using Scrum, and quality assurance directly aligns with the software engineering and computer science tracks within the ECE discipline. The exposure of developing a complete application, from requirement analysis to implementation and testing, provides inuseful insights that complement courses on software engineering, object-oriented programming, and database systems.

To conclude, the industrial training was very relevant to the ECE curriculum. It reinforced classroom learning with hands-on practice, introduced current industry practices aligned with our field, and equipped us with hands-on skills and insights that standard coursework alone could not provide. By experiencing how our knowledge is applied in a real IT company setting, we have acquired a deeper grasp of our field of study and are better prepared for the professional world.To conclude, the industrial training was very relevant to the ECE curriculum. It reinforced classroom learning with hands-on practice, introduced current industry practices aligned with our field, and equipped us with hands-on skills and insights that standard coursework alone could not provide. By experiencing how our knowledge is applied in a real IT company setting, we have acquired a deeper grasp of our field of study and are better prepared for the professional world.

Chapter 2

Organization Profile

In this chapter, the host organization of the industrial attachment **BJIT Limited** is profiled. BJIT Ltd. is a well-known software development and IT services company in Bangladesh with global operations. Understanding the organization's profile is important to appreciate the context in which the training took place, including the company's mission, the services it offers, and its internal structure. The following sections detail the name and overview of the company, its vision and mission, the products and services it offers, and an outline of its organizational structure.

2.1 Name and Overview of the Company/Industry

BJIT Limited (often referred to simply as BJIT) is a prominent global IT consulting and software development firm. The name "BJIT" stands for "Bangladesh-Japan Information Technology", reflecting the company's origins as a joint venture between Bangladeshi and Japanese stakeholders.[3] Established in 2001 as a joint venture between Japanese and Bangladeshi entities, BJIT has grown into a powerhouse in the global IT services market.

(a) BJIT Logo (b) BJIT Academy Logo

Figure 2.1: BJIT logos.

With over 750 skilled engineers and a global presence with offices in Japan, the USA, Sweden, Finland, the Netherlands, Singapore, Thailand, and Bangladesh, BJIT offers high-end IT

solutions including outsourcing, remote development, and comprehensive project management. The company serves a diverse clientele, including over 50 Fortune 500 companies such as Sony, Panasonic, and Google.

Today, BJIT is recognized as an award-winning global IT company with a strong presence in both local and international markets. It employs over 750 skilled engineers and IT professionals and operates out of 8 global offices [4]. This global footprint allows BJIT to serve a wide range of clients across Asia, Europe, and North America. In fact, BJIT has offered services to renowned multinational clients such as Google, Panasonic, Sony, and others [3]. The company's growth and reputation have made it one of the largest offshore software developers in Bangladesh.

2.2 Vision, Mission, and Objectives

Mission: BJIT's mission is to assemble a team of talented people to enhance the client's experience of IT solutions. Another stated mission is to help companies lead change through customized, high-impact software services built for future challenges. The company aims to arise interest from other companies through its expertise in AI, deep learning, financial technology, IoT, and application management software.

Vision: BJIT's ultimate goal is to become the number one IT services company in Bangladesh, with a target of employing more than 10,000 people. A core part of their vision is "growth"—for their clients' businesses, for their engineers, and for all stakeholders involved. BJIT Academy, its educational wing, envisions a landscape where students are nurtured to be globally competent leaders through hands-on, integrated learning experiences. [1]

Core Values/Objectives: The company operates on three core values:

- **Quality is Number One:** Defining quality as solutions that surpass customer needs on time and within budget.
- **Create Customer Value:** Creating value by providing solutions that expand sales, increase operational efficiencies, and reduce costs.
- **Create Stakeholder Value:** Believing the purpose of business is to create value for all stakeholders, including employees and the community.

2.3 Products/Services Offered

BJIT offers a comprehensive suite of software consulting and development services. Their offerings are intended to cater to a wide range of industries, including fintech, e-commerce, transport, and AI. Key services include:

- **Custom Software Development:** This is the core service of BJIT. The company builds software applications tailored to the needs of its clients. These can include web applications, mobile apps, enterprise software systems, and embedded systems. BJIT's engineers

work on various technology stacks (e.g., Java, .NET, C/C++, Python, mobile frameworks) to deliver these solutions. For example, BJIT has experience developing and maintaining large enterprise systems for manufacturing companies, financial institutions, etc., often integrating with legacy systems or building from scratch as required.

- **Outsourced Development & Offshore Centers:** BJIT often acts as an outsourcing partner. Many clients in Japan, Europe, and the USA hire BJIT to be their offshore development team. In such arrangements, BJIT provides dedicated teams that work remotely on the client's projects. This includes handling full software development lifecycle – from requirement analysis, design, implementation, testing to maintenance – under client direction. The company's appeal in this space comes from its cost-effectiveness and high-quality outcomes (leveraging the lower costs in Bangladesh along with disciplined processes). BJIT is described as a "Trusted Partner" for outsourcing, known for exceeding expectations in quality [4].

- **AI and Deep Learning:** Developing intelligent systems using machine learning, with a focus on computer vision, audio, and NLP.

- **IoT and Embedded Solutions:** Building systems that interact directly with hardware for industrial applications, smart devices, and automotive components.

- **QA and Test Automation:** Ensuring software quality through rigorous manual and automated testing procedures.

- **DevOps and Cloud Services:** Assisting businesses with cloud migration, infrastructure management, and implementing CI/CD pipelines across AWS, Azure, and Google Cloud.

- **Training and Capacity Building:** Through BJIT Academy, the company also offers training services. While this is more of an internal product aimed at building a pipeline of skilled engineers, it can be considered a service to the broader industry by increasing the skill level in the region. BJIT Academy runs regular training programs for fresh graduates (as noted, a four-month intensive program where many graduates join BJIT upon completion [1]).

- **Enterprise Solutions:** Expertise in ERP and CRM platforms like SAP, Salesforce, and Odoo.

BJIT has also developed its own products, such as **miniERP**, **Xamify** (a computer-based testing tool), and **FaceAI** (a face recognition SDK). In summary, BJIT's products and services cover **end-to-end software solutions**. Whether a client needs a small mobile app or a large-scale enterprise system, BJIT has the capability to deliver. They can function as an extension of a client's in-house team (outsourcing/offshore service) or deliver turnkey projects. This diversity of services provided an enriching backdrop for our internship, as we could get exposure to

multiple facets of the software industry – from development to testing to consulting on best practices.

2.4 Organizational Structure

BJIT Group has a global and diversified organizational structure, with its main development hub in Dhaka, Bangladesh, and multiple offices worldwide to serve its international clientele. The group consists of several entities, including BJIT Ltd., BJIT Inc. (Japan), BJIT Corp (USA), BJIT Oy (Finland), and others. The leadership team comprises exposured executives from global technology giants.

The key leadership includes:

- **Akbar JM:** Founder, Chairman & CEO.
- **Masaki Horikawa:** CEO, BJIT Inc.
- **Mehedi Masud:** COO, BJIT Ltd. & CEO, BJIT HR Agency.
- **Abul Kalam:** President & CTO, BJIT Ltd.
- **Homayun Kabir:** Head of Operation, BJIT Academy.

The company is organized into several key divisions, including the Engineering Division, Sales and Marketing, Administration, and Human Resources, which work collaboratively to deliver projects.

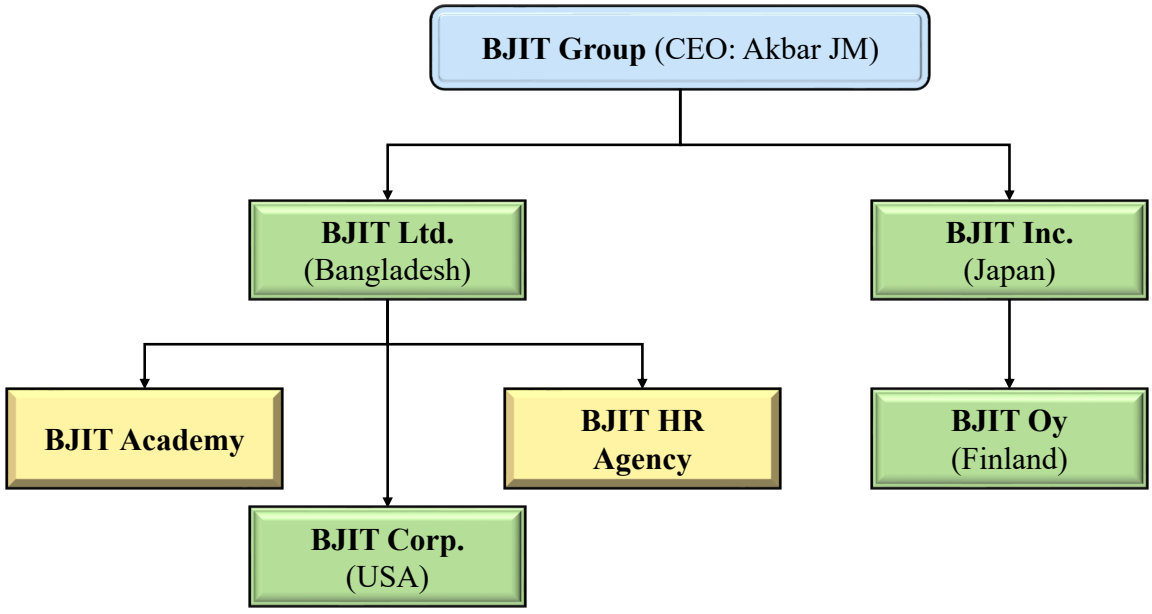

Figure 2.2: Simplified Organizational Structure of BJIT Group.

Chapter 3

Training Activities

In this chapter, we provide a comprehensive overview of the training activities during the two-week industrial attachment at BJIT. It covers the various departments and sections we engaged with, the tasks and assignments we were given, the skills and methods we learned, the tools and software we used, and the safety or professional practices we followed. The training was well-structured to progress from introductory sessions into hands-on project work, interspersed with expert talks and soft skills workshops. Below, each aspect of the training activities is detailed.

3.1 Departments/Sections Where Training Was Conducted

The entire two-week industrial attachment was conducted under the supervision of **BJIT Academy**, the specialized training and development wing of BJIT Ltd. BJIT Academy is responsible for nurturing talent and equipping engineers with industry-relevant skills. The training was held at their facility at BJIT Academy, Level 1, and was structured to provide a comprehensive overview of industry practices, from software development methodologies to practical coding experience and soft skills development.

3.2 Description of Tasks Assigned

The program was intensive and meticulously planned over ten working days, from March 3 to March 14, 2025. The activities were a blend of theoretical sessions, hands-on development work, and interactions with industry experts. The daily tasks were as follows:

- **Week 1 (March 3 - 7):**
  - **Day 1:** Introduction to the team, scope of the training, and an overview of BJIT.
  - **Day 2:** Sessions on Software Development Process & Methodologies (SDLC) and tools used in the industry. Practical experience kick-off with requirement analysis (SRS) and system design.

- **Day 3:** Environment setup for development work. Introduction to Scrum methodology, including forming a Scrum team and practicing its activities. Session on Software Domains and Technology Trends.
- **Day 4:** Focused project development work. Sessions on manual SQA and soft skills (Communication).
- **Day 5:** Continued development work. Sessions on high-performing team workplace ethics and SQA Automation from an industry expert.

- **Week 2 (March 10 - 14):**
  - **Day 6:** Continued project development. Session on Technology Introduction and Trends.
  - **Day 7:** A session on Artificial Intelligence (AI) from an industry leader, followed by development work.
  - **Day 8:** Final development work and submission of assigned tasks.
  - **Day 9:** Demo of the development work completed during the training.
  - **Day 10 (Final Day):** Networking opportunities, including a visit to the engineering division and interaction with different development teams. A Q&A session, followed by a closing ceremony and photo session.

3.3 Skills and Techniques Learned

The industrial attachment was very educational, considerably enhancing both our technical and soft skills. Through the various training activities and tasks described, we developed numerous skills and learned modern techniques. The key skills and methods acquired include:

- **Technical Skills:**
  - **Python Programming:** Although we had prior knowledge of Python basics, this training solidified and expanded our Python programming skills, we gained proficiency in Python fundamentals, including data types, control flow, functions, and object-oriented programming (OOP) concepts. We learned to write more efficient code, handle file I/O using the json module, and manage errors gracefully. For example, by implementing the task manager, we practiced using Python data structures (lists, dictionaries) in practical scenarios and writing functions with clear purpose. We also learned new Python techniques like using list comprehensions for filtering tasks and the datetime module for handling dates.
  - **Object-Oriented Programming (OOP):** Learned to apply core OOP principles like Encapsulation, Inheritance, Polymorphism, and Composition through practical assignments.

    - *Encapsulation:* Keeping class internals hidden and exposing operations (e.g., book borrowing logic inside class).
    - *Inheritance:* Reusing Book attributes in EBook and extending functionality.
    - *Polymorphism:* Though our scenario was simple, the concept of overriding `borrow_book()` in EBook to behave differently (limit borrow days) illustrated polymorphism.
    - *Composition:* The Library class managing collections of books/patrons showed how classes collaborate. Designing and coding this system improved our ability to translate real-world problems into software models, a critical software engineering skill.
  - **Software Development Methodologies:** Acquired a practical understanding of the SDLC, Agile principles, and the Scrum framework.
  - **Software Quality Assurance (SQA):** Learned the distinction and application of manual and automation testing techniques.
  - **Data Handling:** Gained experience with data persistence using the JSON module for saving and loading application data.

- **Professional Skills:**
  - **Teamwork and Collaboration:** Practiced working in a team environment by forming a Scrum team and collaborating on project tasks.
  - **Communication:** Enhanced communication skills through presentations (project demo) and interactions with mentors and industry leaders.
  - **Professional Ethics:** Gained insights into workplace ethics and the dynamics of high-performing teams.
  - **Problem-Solving:** Developed analytical and problem-solving skills by tackling various programming assignments with specific constraints and requirements.

3.4 Tools, Software, or Machinery Used

The training was primarily software-focused. The main tools and tech used during the attachment include:

- **Programming Language:** Python 3.x.
- **Integrated Development Environment (IDE):** Visual Studio Code (or a similar text editor like Sublime Text, PyCharm).
- **Version Control:** Git and GitHub for code management (inferred as a standard industry practice).

- **Data Interchange Format:** JSON for data persistence in the Task Manager project.
- **Operating System and Environment:** Our training took place on Windows 10 based PCs (which were provided by BJIT in their training lab). We gained comfort in using the command prompt/PowerShell for running Python scripts and Git commands.
- **Collaboration Tools:** Communication platforms like Slack or Microsoft Teams for team coordination (inferred).
- **Standard Office Software:** Microsoft Office Suite (Word, Excel, PowerPoint) for documentation and presentations.

In essence, the tools and tech we used or learned about gave us a mini-replica of a modern development environment. We started from writing code in an IDE, using version control for collaboration, to eventually presenting our work with documentation and diagrams.

This tooling exposure also taught us the importance of choosing the right tool for a task: a simple text file (JSON) was sufficient for our database needs; a high-level language like Python accelerated development; version control is non-negotiable for collaboration; and using an IDE increased our productivity versus a basic text editor. These are insights that will guide our approach to projects going forward.

3.5 Safety Practices Followed

While the training was in a software development environment with minimal physical hazards, a strong emphasis was placed on **information security**. A dedicated training and assessment program on Information Security was conducted, for which a certificate of completion was awarded. This training covered the importance of protecting company data, client information, and intellectual property. Key practices included secure coding habits, understanding data privacy, and adhering to the company's IT policies.

To summarize, even though our work was on computers in an office, safety was never overlooked. The internship ingrained in us that being a good engineer isn't just about writing code; it's also about being trustworthy with data, being responsible in actions, communicating respectfully, and looking after one's health and surroundings. These practices ensure that the workplace remains safe, inclusive, and productive for everyone

Chapter 4

Project and Assignment Work

During the industrial attachment, we engaged in multiple assignments and a capstone project which collectively reinforced our learning. In this chapter, we detail the specific projects and assignments handled (what they were and their scope), the methodology we used in executing them, the outcomes or results we achieved, and the challenges we faced along with the solutions we applied. The structured approach to these tasks demonstrates how we applied theoretical knowledge to hands-on problems and adapted to real-world constraints.

4.1 Specific Project(s) or Assignments Handled

During the training, we were tasked with several assignments of increasing complexity, culminating in a final project. These assignments were intended to reinforce the concepts taught during the sessions.

1. **Practice 1: Array Maximum Difference:** A fundamental problem-solving task to find the maximum absolute difference between any two elements in a user-provided array.

2. **Practice 2: Calculate Total Cost:** A dictionary-based problem to calculate the total cost of fruits from two dictionaries, one containing prices and the other quantities, with specific conditions for handling missing items and zero quantities.

3. **Final Individual Assignment: Library Management System:** A comprehensive project to demonstrate OOP principles. The task was to build a system to manage books and patrons, with functionalities for borrowing and returning. It required implementing classes for ‘Book‘, ‘Patron‘, and ‘Library‘, and also demonstrating inheritance by creating an ‘EBook‘ subclass.

4. **Group Project(Final Project)- Task Manager with Analytics:** A command-line application to manage tasks. The requirements included adding, removing, updating, and listing tasks, persisting data to a JSON file, and displaying analytics like the number of completed and overdue tasks. The project integrated multiple features:

- Data model for tasks (list of task dictionaries)
- o Functions to add/update/remove/list tasks
- An interactive CLI menu for user input
- Analytics computation (total tasks, completed, pending, overdue)
- JSON file read/write for saving tasks between sessions
- (Optional) a search feature for tasks by keyword

This project was tackled by the team of interns collaboratively and spanned the majority of the internship period.

Roles and Contributions: For the group project, we loosely assigned roles to coordinate work: One member acted as a Project Lead/Scrum Master, one member was the Lead Developer on the analytics part (writing most of the analytics logic). One member focused on the Persistence (JSON I/O). Others took charge of Task CRUD functions and Menu interface. We all contributed to testing and integration.

Relevance of Each: The practice assignments were directly tied to Python basics and thus built a foundation for the more complex tasks. - The library OOP assignment was somewhat independent of the main project but reinforced programming discipline and design thinking. - The Task Manager project was the most relevant to the real world – akin to a small software project you might do in a company. It tied together everything: analysis, design, coding, testing, teamwork, and presentation.

In essence, over the two weeks, we handled two small coding assignments, one larger programming assignment (OOP), and one team software project. This multi-faceted workload made sure a well-rounded exposure.

4.2 Methodology Used

The development approach for the assignments followed the principles taught during the training. For the larger projects (Task Manager and Library Management System), a mini-SDLC approach was adopted:

1. **Requirement Analysis:** Carefully studying the problem statements and requirements provided in the assignment documents. For instance, in the Library Management System, this involved identifying the attributes and methods for each class (‘Book‘, ‘Patron‘, etc.).

2. **System Design:** Planning the structure of the application. For the OOP-based project, this meant designing the classes, their relationships (inheritance and composition), and the flow of logic. For the Task Manager, this involved designing the functions for each feature and the structure of the JSON data.

3. **Implementation (Coding):** Writing the Python code to implement the designed logic. This was an iterative process involving writing functions and classes, and testing them incrementally.

4. **Testing:** Manually testing the application with various inputs to ensure all requirements were met and to identify bugs. For example, testing the Task Manager with edge cases like updating a non-existent task or filtering by an invalid status.

5. **Deployment:** The ”deployment” in this context was the final submission of the source code and a demonstration of the working application.

Overall, while the scale was small, we effectively employed a hybrid methodology combining agile practices with structured programming techniques.

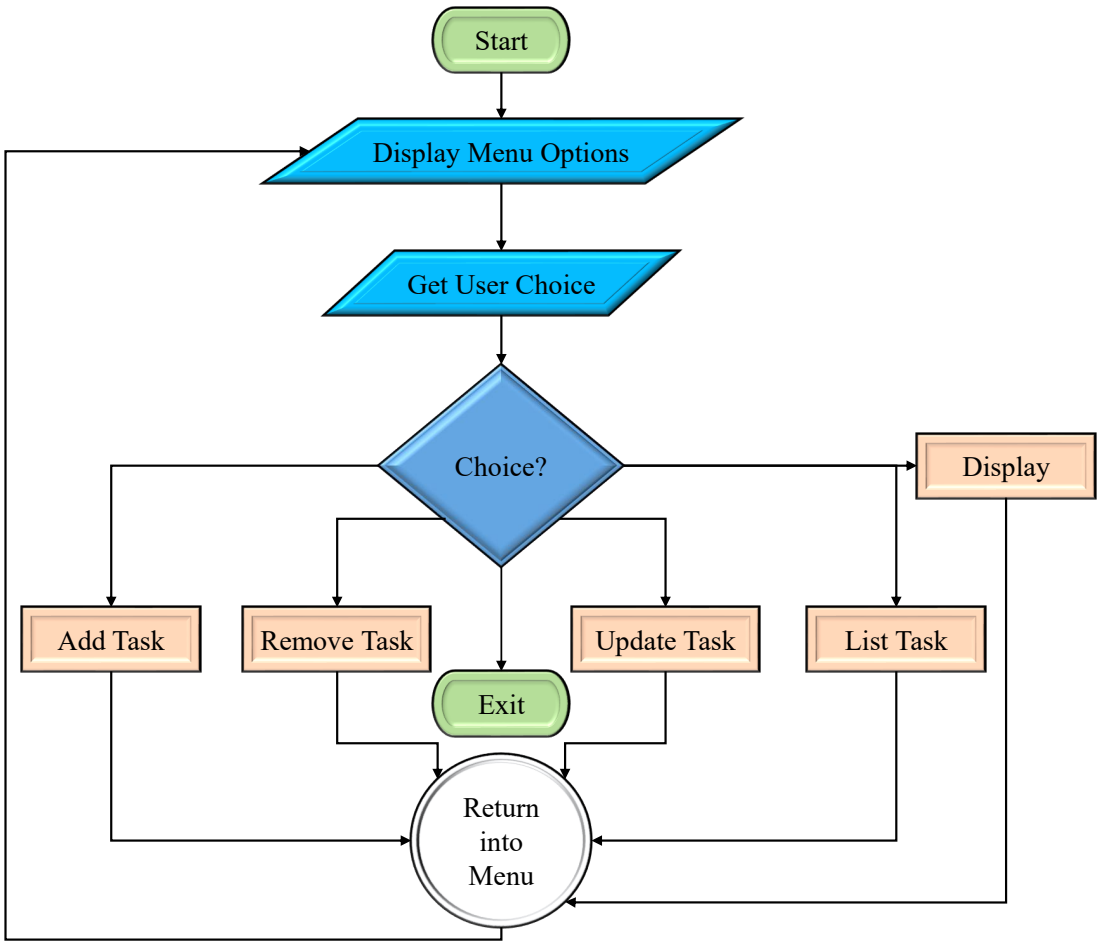

Figure 4.1: Flowchart for the Task Manager Application Menu.

4.3 Outcomes or Results Achieved

All assigned projects were successfully completed and showed on the final day.

- Assignment 1 **(Array Max Difference)**: We successfully wrote and executed a Python program that reads an integer n and n array elements, then prints the maximum absolute difference. The expected result (and our program's output) for the test case given (with array elements like 10 and -3) was 13, which our program produced correctly[25]. The code was concise (leveraging either sorting or single-pass logic) and demonstrated correct handling of both positive and negative values. This outcome confirmed our understanding of basic Python and problem-solving logic. We documented a sample run in our report as proof of correctness:
  - Example: Input: 5 (for n), followed by -3 0 7 10 5 as array elements;
  - Output: 13.
- The **Library Management System** successfully demonstrated all the required OOP concepts. The classes were implemented with their respective attributes and methods. The system correctly handled borrowing and returning books, preventing a borrowed book from being borrowed again. The use of inheritance (‘EBook‘ subclass) and composition (‘Library‘ class managing ‘Book‘ and ‘Patron‘ objects) was successfully implemented and showcased.
- The **Task Manager** application was fully functional, allowing users to perform all CRUD (Create, Read, Update, Delete) operations on tasks. The data persistence using JSON worked correctly, and the analytics feature provided accurate counts of total, completed, pending, and overdue tasks. We delivered a fully functional command-line application. The key outcomes:
  - *Task Creation/Management:* Users can add new tasks with title, description, status, creation time, and optional due date. Removal and updating of tasks also function as intended.
  - *Listing Tasks:* The application lists all tasks, and we implemented filtering by status.
  - *Analytics:* The app displays a summary showing total number of tasks, number completed, number pending, and number overdue.
  - *Persistence:* Perhaps the most satisfying outcome was the persistence – tasks saved to a JSON file.
  - *User Interface:* The CLI menu interface is user-friendly, prompting users for choices and guiding them through inputs. We handled invalid inputs by showing an error and re-displaying the menu, ensuring the program doesn't crash on wrong usage.

- *Search Feature:* As an extra, our application can search tasks by a keyword. This returns tasks whose title or description contains the keyword, making it easier to find tasks in a long list. We tested this by adding tasks with distinct keywords and searching.
- *Quality of Code:* Though not visible to end-user, our code quality improved through the project. We structured it into functions and kept it relatively modular.

The project was showcased in a presentation to BJIT's team, and the **feedback was very positive.** They noted that in just two weeks, producing a working tool with multiple features is commendable. They appreciated the inclusion of analytics and persistence, as it showed understanding beyond trivial scripting.

The source code for these projects is included in the Appendices section of this report.

4.4 Challenges Faced and Solutions Applied

Several challenges were encountered during the project work, which offered useful learning exposures.

- **Challenge:** Implementing the logic for handling overdue tasks in the Task Manager required careful handling of date and time formats. Converting string-based dates to datetime objects for comparison was initially tricky.

  Solution: The `datetime` module in Python was studied and used to parse the date strings and compare them with the current date, successfully implementing the overdue task detection.

- **Challenge:** In the Library Management System, ensuring that the state of a `Book` object (e.g., `is_borrowed`) was correctly updated and reflected across the system when a `Patron` borrowed or returned it was a key challenge related to object interaction.

  Solution: This was solved by passing book objects by reference. The `Patron`'s `borrow_book` method called the `Book`'s own `borrow_book` method, ensuring the original book object's state was modified, thus maintaining data integrity across the system.

- **Challenge:** Managing the project within the tight deadlines of the two-week training program was demanding.

  Solution: Applying the time-boxing and prioritization principles learned from the Scrum sessions helped in breaking down the work into smaller, manageable tasks and focusing on delivering the core requirements first. Regular check-ins with team members and mentors also helped in staying on track.

Chapter 5

Observations & Learning Outcomes

In this chapter reflects on the key observations made during the training and enumerates the learning outcomes. The industrial attachment offered insights that went beyond technical skills; it offered a window into the workings of a software company, highlighting differences between academic learning and industry practice. We categorize the learning outcomes into technical knowledge acquired, exposure to the industry environment, improvements in teamwork and professional skills, and grasp the gap between theoretical knowledge and hands-on application.

5.1 Technical Knowledge Gained

The industrial attachment at BJIT was a period of immense technical learning. The hands-on approach solidified many theoretical concepts and introduced new industry-standard practices.

5.1.1 Software Development Life Cycle (SDLC)

A significant learning outcome was the hands-on grasp of the Software Development Life Cycle (SDLC). We learned that SDLC is a systematic framework for managing a software project, ensuring quality and efficiency from start to finish.[5] The typical phases we were introduced to were:

1. **Planning & Requirement Analysis:** Gathering requirements from stakeholders to define project goals.
2. **Design:** Creating the software architecture and system design based on the requirements.
3. **Implementation:** Writing the code for the product.
4. **Testing:** Verifying the software against requirements to find and fix defects.
5. **Deployment:** Releasing the software to users.
6. **Maintenance:** Supporting and improving the software after release.

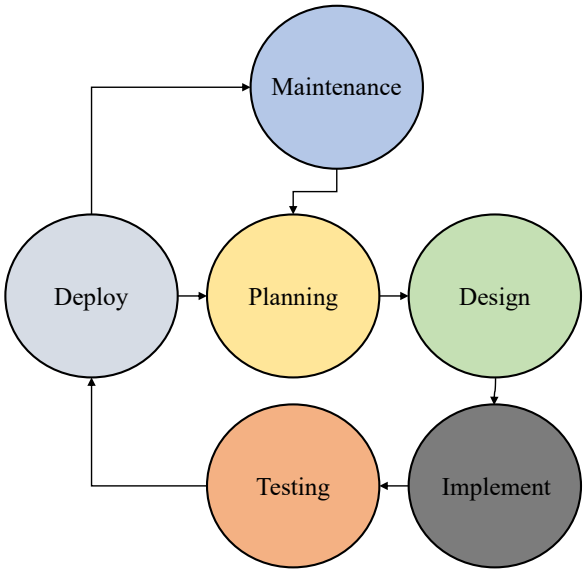

Figure 5.1: Iterative Software Development Life Cycle (SDLC).

5.1.2 Agile and Scrum Methodology

We were introduced to the Agile methodology, which emphasizes iterative development, collaboration, and flexibility. Within Agile, we focused on the **Scrum** framework. Key concepts learned include [6]:

- **Scrum Roles:** Product Owner, Scrum Master, and the Development Team.
- **Scrum Events:** Sprints (short, time-boxed cycles), Daily Stand-ups, Sprint Planning, and Sprint Retrospectives.
- **Scrum Artifacts:** Product Backlog, Sprint Backlog, and the Increment.

The exercise of forming a Scrum team and practicing these events gave a real sense of how modern software teams operate to deliver value incrementally and respond to change effectively.

5.1.3 Software Quality Assurance (SQA)

The sessions on SQA were eye-opening. We learned about the critical role of testing in ensuring software quality. The key distinction between manual and automation testing was highlighted:

- **Manual Testing:** Performed by a human tester, it is essential for exploratory testing, usability testing, and assessing user experience. It relies on human intuition and creativity but can be slow and prone to error.[7]
- **Automation Testing:** Uses scripts and tools to execute tests. It is ideal for repetitive tasks like regression testing, load testing, and performance testing. It is faster, more reliable, and increases test coverage but requires an initial investment in tools and script development.[7]

We learned that a hybrid approach, combining the strengths of both, is often the most effective strategy in the industry.[8]

5.1.4 Proficiency in Python Programming

Our comfort and skill in Python increased considerably. We progressed from writing simple scripts to structuring a multi-module Python project. We learned about Python-specific features such as:

- Using built-in data structures effectively (list, dict).
- File handling and the json module for serialization.
- Error handling using try-except blocks.

5.1.5 Data Formats and File Handling

We specifically learned about JSON as a data format – its syntax, flexibility, and how hands-only every language can parse it. We saw why JSON is popular for persisting structured data in a human-readable way. This knowledge is useful for tasks like reading configuration files or writing simple data interchange for small tools. Moreover, file I/O basics (open/read/write in Python) are now firmly understood.

5.1.6 Soft Technology Skills

Another kind of technical knowledge is using professional tools – not just coding but tools to aid coding:

- We became adept at using VS Code and customizing it (installing extensions, using its debugger, etc.).
- We learned to use Slack for communication effectively, which is a technical skill in workplace communication tools.
- Using draw.io to create UML and flowcharts improved our ability to convey technical information visually.
- Utilizing cloud repositories (GitHub) introduced us to remote collaboration concepts like pull requests (we browsed a bit, though we merged directly, we saw how PRs could work by looking at an example on an open-source repo).

Overall, the technical knowledge acquired is quite expansive for a short period. We essentially took a deep dive into software development with a hands-on project and emerged with a clearer, more integrated grasp of how technologies and practices fit together. We can code better, design better, test better, and use tools better. This strong technical foundation and the exposure to modern practices (like agile, version control, etc.) have prepared us to take on more complex projects in the future, whether in academic projects or professional roles.

5.2 Practical Exposure to the Industry Environment

The training offered a simulated but realistic glimpse into the professional software industry. The emphasis on deadlines, structured daily schedules, and clear deliverables for assignments mirrored a real work environment. Interacting with senior industry leaders from BJIT during sessions on AI, SQA, and technology trends offered useful insights into the current state and future direction of the tech industry. The final day's networking session and visit to the engineering division enabled us to see professional developers at work and understand the dynamics of different teams. Some key observations and learnings from the industry environment exposure include:

- **Team Dynamics and Culture:** We observed how software teams operate in a professional setting. At BJIT, the culture was collaborative and relatively informal in communication, yet professional.
- **Work Discipline & Ethics:** By being in the office, we learned the importance of punctuality – work started at a set time (we had to be in by 9:00 am for sessions). People took timely lunch and tea breaks and then returned to work, maintaining a steady routine. There was a sense of accountability; if someone was stepping out or unable to meet a commitment, they informed their team. We adopted these habits.
- **Agile Work Environment:** Seeing Scrum in action was eye-opening. We participated in a simulated stand-up meeting each morning for our team, but we also witnessed a real Scrum team's stand-up.
- **Multidisciplinary Interactions:** We got to interact with professionals of various domains: developers, QA engineers, UX designers (one gave us tips on how to make CLI output more user-friendly), and even people from business development during the networking session. This exposure taught us that a software company is not just about programmers typing away; it's a symphony of multiple roles. These interactions underscored the importance of each role and how they interconnect, something we seldom think about in academia (where focus is mostly on the developer role).

5.3 Teamwork, Communication, and Professional Ethics Learned

Beyond technical skills, the training heavily emphasized soft skills.

- **Teamwork:** The Scrum team formation was a practical lesson in collaboration. We had to coordinate tasks, help each other with debugging, and work towards a common goal for the project demo.
- **Communication:** The soft skills session on communication and the requirement to present our project demo helped improve our presentation and technical communication abilities.

- **Conflict Resolution:** Minor disagreements did occur (e.g., about how to implement a feature). We learned to resolve these by focusing on objective criteria (what meets requirements best, what's simplest for now) rather than personal opinions.
- **Professional Ethics:** The session on "High Performing Team Workplace Ethics" provided guidance on professionalism, responsibility, and integrity in the workplace. The information security training also underscored the ethical responsibility of handling data securely.
- **Accountability:** We held each other accountable gently – if someone was falling behind on a task, we'd check in and see if they needed help or more time, rather than ignoring it. This peer accountability ensured we met our deadlines.
- **Presentation Skills:** Preparing and delivering the final presentation honed our ability to communicate with a larger audience. We learned to maintain eye contact, modulate voice, and gauge audience reactions. The positive reception of our presentation was a testament to improvement in our public speaking.
- **Interpersonal Communication:** Simply being around professionals, we picked up on polite conversational norms. We learned how to introduce ourselves to new people, how to network by asking about their role and showing interest, and how to maintain a respectful tone even in casual chats.

5.4 Gap Between Theory and Practice

This training was instrumental in highlighting the gap between academic learning and industry practice.

- **Scale and Complexity:** Academic projects are often small and well-defined, whereas industry projects are large, complex, and involve collaboration between multiple teams.
- **Software Development Process:** Academically, we learn various models (Waterfall, Agile, etc.) often in a somewhat idealized form. In practice, we observed a hybrid approach. For example, while BJIT teams followed Agile/Scrum, they still had documentation and planning steps reminiscent of Waterfall for initial phases.
- **Importance of Methodologies:** While we learn programming languages in university, the structured approach of using SDLC and Agile/Scrum to manage projects is something that can only be truly appreciated in a practical setting.
- **Code Quality and Maintenance:** In academia, the focus is often on getting the correct output. In the industry, there is a huge emphasis on writing clean, maintainable, and well-documented code, as it will be read and modified by many others over its lifetime.

- **Learning New Technologies:** University courses often have fixed syllabi and we master certain languages or tools. In practice, we saw that engineers constantly learn on the job. During the two weeks, we ourselves had to learn things not explicitly taught before (like using Git, or the specifics of JSON and web-related topics from sessions). This proved that self-learning and quick adaptation is a crucial skill. The gap here is that in theory, you might think you'll use only what you formally learned, but in practice, technologies evolve and vary by company – one must be ready to learn new things rapidly.

- **The Role of Testing:** SQA is a vast and critical field that is often only touched upon lightly in academic courses. The training revealed its central importance in the development process.

- **Quality vs. Delivery Pressure:** Theoretically, we are told to always strive for highest quality. In practice, there is sometimes tension between meeting a deadline and polishing something further. We saw that in the professional environment, quality is extremely important (especially for delivered products), but there are pragmatic decisions like deferring a minor feature or known low-priority bug to a later patch to meet a release date. We experienced a tiny version of this when we postponed the optional search feature implementation to after we had finished core features. The key is to maintain quality on what's delivered and plan improvements.

In summarizing the gap between theory and practice: Theory gave us the language and baseline (we wouldn't have thrived in the internship if we didn't know programming, data structures, algorithms, etc.). Practice taught us context, scale, and the need for adaptability and pragmatism.

Chapter 6

Problems Encountered & Solutions

The two-week industrial attachment, while immensely beneficial, was not without its challenges. Overcoming these hurdles was a significant part of the learning process.

6.1 Technical Challenges

- **Environment Setup:** On the first day of development work, some time was spent ensuring that everyone had the correct version of Python and the necessary IDE extensions installed. Minor discrepancies in environments led to initial confusion. **Solution:** The mentors from BJIT Academy provided clear instructions and hands-on assistance to resolve these setup issues quickly, ensuring everyone had a consistent development environment.

- **Debugging Complex Logic:** In the "Task Manager with Analytics" project, implementing the logic to correctly identify overdue tasks while handling various date formats and potential user input errors was challenging. A simple string comparison was not sufficient.

  Solution: This required a deeper dive into Python's `datetime` module. Through documentation and mentor guidance, I learned to parse date strings into datetime objects, handle potential `ValueError` exceptions for invalid formats, and correctly compare them with the current date to filter overdue tasks.

- **Grasping OOP Concepts Practically:** While OOP concepts were familiar from coursework, applying them to design the Library Management System from scratch was a challenge. Deciding which attributes should be private (encapsulation) and how classes should interact (composition) required careful thought.

  Solution: The problem was broken down into smaller parts. I started by implementing the `Book` class, then the `Patron` class, and tested their interactions before integrating them into the main `Library` class. This modular approach, combined with reviewing the lecture slides and seeking clarification from instructors, made the implementation manageable.

- **Performance Consideration:** Although our application was small-scale, we did consider the performance of certain operations. For example, listing or searching tasks in a list is O(n), which is fine for n small but we pondered if n were large.

  Solution: We discussed data structure choices. We considered using a dictionary for tasks keyed by title for faster removal and search. We decided against it for now because our titles might not be unique or might need multiple tasks with same title (which our spec didn't forbid), but the exercise of thinking about it was valuable. We realized our education in algorithm complexity is important, and we concluded that for our expected usage (¡100 tasks perhaps during our tests) performance is a non-issue. However, we left a comment in code noting that if extended to many tasks, one might use an index or dict for faster lookups. This challenge was more hypothetical, but it was good to apply theoretical analysis to our practical design and consciously make a decision.

6.2 Administrative/Logistic Issues

- **Intensive Schedule:** The training schedule was packed with sessions and development work, leaving limited time to absorb and reflect on each topic in depth. The pace was fast, especially for complex topics like SQA Automation and AI.

  Solution: I focused on taking detailed notes during the sessions and utilized the Q&A portions to clarify doubts immediately. I also dedicated time after the official training hours to review the concepts and work on the assignments, which helped in keeping up with the pace.

- **Short Duration:** A two-week period is very short for a comprehensive industrial attachment. While it provided an excellent overview, it was not enough time to delve deep into any single technology or participate in a live, ongoing industry project.

  Solution: I maximized the learning within the given timeframe by being proactive, asking questions, and engaging fully in all activities. I also made sure to connect with the industry experts and mentors to understand the broader context of their work, which provided insights beyond the scope of the training modules.

- **Resource Constraints:** At times, there was contention for resources like the internet bandwidth or even mentors' time. One afternoon, we had trouble downloading a required library (though mostly we worked offline, at one point we considered using a Python library for colored output which required pip installing a package).

  Solution: We circumvented heavy dependency by writing a simpler solution ourselves (for color, we manually inserted ANSI codes instead of using an external library). For internet, we did downloads in off-peak times once we realized lunch hour was slower due to many using network. Also, when we needed mentor guidance and they were busy, we

pooled questions and asked in one sitting rather than frequent interruptions. Essentially, we optimized our usage of available resources and found alternatives when something wasn't readily available.

- **Balancing Training Sessions with Project Work:** The first week had many sessions, leaving limited contiguous time for coding. This fragmentation was a logistical challenge as deep work on coding got interrupted.

  Solution: We made the most of the time after official hours and a bit on the weekend. We got permission to use the training room on the Saturday in between for a half day. By doing so, we caught up on the project tasks. Also, we each sometimes coded small pieces at home after hours.

6.3 How These Were Resolved or Managed

The resolution of these challenges was a testament to the supportive environment created by BJIT Academy and the collaborative spirit among the trainees.

- **Mentor Guidance:** The instructors and mentors were always approachable and willing to help with both technical and conceptual problems. Their guidance was the primary resource for overcoming complex technical hurdles.
- **Peer Collaboration:** Working alongside fellow trainees was incredibly helpful. We often discussed problems, shared solutions for debugging, and explained concepts to each other, which reinforced our own understanding.
- **Proactive Learning:** I took the initiative to read documentation and search for solutions online when faced with a problem. This self-learning approach was crucial for solving specific coding issues and for deepening my understanding of the tools and technologies being used.

By managing these problems, we made sure the internship stayed on track and its objectives were met. Importantly, each problem we encountered and solved gave us confidence and a learning exposure on how to handle similar issues in the future. In a sense, facing these problems was itself a useful part of the training, as it simulated real-life project hurdles and taught us the resilience and skills needed to overcome them.

Chapter 7

Conclusion

The industrial attachment at BJIT Ltd. was an enriching exposure that considerably contributed to my growth as a budding engineer. Over the two-week period, I had the opportunity to apply academic knowledge in a hands-on environment, develop new skills, and gain insights into professional software development practices. This concluding chapter summarizes the overall training exposure, reflects on how it has contributed to my academic and professional development, and discusses the future scope or career relevance of the skills and exposures acquired.

7.1 Summary of the Overall Training Experience

The two-week industrial attachment at BJIT Ltd. was an inuseful and enriching exposure. It offered a comprehensive and hands-on overview of the software development industry, from high-level methodologies to hands-on coding. The program was exceptionally well-structured, balancing theoretical knowledge with hands-on application through a series of challenging assignments and projects. The exposure to industry experts, the emphasis on soft skills, and the simulation of a professional work environment have collectively offered a holistic learning exposure that extends far beyond the confines of a traditional classroom. I leave this training with a clearer grasp of the industry's expectations and a renewed enthusiasm for software engineering.

7.2 How It Contributed to Academic and Professional Development

This training has considerably contributed to both my academic and professional growth.

- **Academic Development:** It provided a practical context for the theoretical knowledge gained in courses like Object-Oriented Programming, Data Structures, and Software Engineering. Seeing concepts like inheritance, polymorphism, and SDLC methodologies applied to solve real problems has deepened my understanding and appreciation of these subjects.

- **Professional Development:** The attachment served as a crucial stepping stone into the professional world. I have gained hands-on experience with industry-standard tools and practices, developed essential soft skills like teamwork and communication, and built a foundational understanding of corporate culture and professional ethics. This experience has boosted my confidence and made me better prepared for future career opportunities.

Overall, the attachment has made me more competent and confident academically (in tackling complex projects and grasp course materials deeply) and professionally (in possessing the hands-on skills and attitude needed in the workplace).

7.3 Future Scope or Career Relevance

The skills and knowledge acquired during this training are directly relevant to my future career aspirations in the field of software development. The proficiency acquired in Python, a versatile and widely used language, opens up opportunities in various domains, including web development, data science, and artificial intelligence. The hands-on exposure with the Agile and Scrum frameworks is particularly useful, as these are the dominant methodologies used in the modern tech industry. Additionally, the insights acquired from sessions on emerging technologies like AI have offered a glimpse into future career paths and areas for further specialization. This training has not only equipped me with hands-on skills for an entry-level software engineering role but has also offered a clearer roadmap for my long-term career development.

To conclude, the industrial attachment was not an isolated academic exercise, but a pivotal step towards my professional life. It validated my career choice, equipped me with hands-on tools and soft skills, and clarified what it takes to succeed in the tech industry. The knowledge and exposure acquired will be directly relevant as I pursue job opportunities after graduation and will likely accelerate my onboarding and performance in any organization I join. Moreover, it has inspired me to continue growing – both by leveraging what I learned and by seeking new learning opportunities to further narrow the gap between being a student and becoming a fully-fledged engineering professional.

Chapter 8

Recommendations

Based on my exposures during the industrial attachment, I have formulated several recommendations. These are directed toward two main audiences: firstly, suggestions for improving the training program itself for future batches (drawing from what worked well and what could be enhanced in our internship at BJIT), and secondly, personal recommendations for future trainees (we who will undertake similar industrial attachments). The goal is to help maximize the benefits of such programs for all stakeholders.

8.1 Suggestions for Improving the Training Program

While the training program was excellent, a few adjustments could potentially enhance the learning exposure even further.

- **Extended Duration:** The two-week duration, though intensive, feels very short for the breadth of topics covered. Extending the program to three or four weeks would allow for a deeper dive into key areas like SQA automation or AI and would provide more time for the final project, possibly allowing for more complex features.
- **Inclusion of a Mini Live Project:** While the assignments were very practical, involving trainees in a small, non-critical part of a live, ongoing project at BJIT could provide an unparalleled experience of real-world development workflows, version control collaboration, and code reviews.
- **More Focused SQA Automation Session:** The session on SQA automation was very insightful. A hands-on workshop where trainees could write a simple test script using a framework like Selenium or PyTest would be a valuable addition to complement the theoretical overview.
- **Real-world Project Samples or Code Reading:** While building our own project was great, it might be useful to also have an exercise where interns read or contribute to an existing codebase (perhaps an open-source project or a simplified internal project of

BJIT). This teaches how to navigate and understand someone else's code – a valuable skill.

- **Dedicated Session on Version Control:** Although inferred as a standard practice, a dedicated hands-on session on Git and GitHub, covering branching, merging, and pull requests, would be extremely beneficial for students who have limited prior experience with collaborative version control.

8.2 Personal Recommendations for Future Trainees

To the we who will undertake this industrial training in the future, I would like to offer the following advice to make the most of the opportunity:

- **Strengthen Your Fundamentals:** Before starting the training, revise your basic programming concepts, especially in the language that will be used (e.g., Python), and have a clear understanding of Object-Oriented Programming principles. This will allow you to focus on the new, industry-specific concepts being taught.

- **Be Proactive and Ask Questions:** The mentors and industry experts are a wealth of knowledge. Do not hesitate to ask questions, no matter how simple they may seem. Your engagement and curiosity will significantly enhance your learning.

- **Collaborate with Your Peers:** Work closely with your fellow trainees. Discussing problems and sharing knowledge is one of the most effective ways to learn. You are a team, not competitors.

- **Focus on Understanding, Not Just Completing:** The goal is not just to finish the assignments but to understand the underlying concepts. Take the time to understand why a particular solution works and how it relates to the principles of good software design.

- **Network Effectively:** Use the opportunity to connect with the professionals at the company. Learn about their career paths, their experiences, and the challenges they face. These connections and insights can be invaluable for your future career.

Chapter 9

References

[1] Marubeni Corporation, *Next generation — #10 bjit — scope*, https://www.marubeni.com/en/brand_media/scope/bjit/, Accessed: 2025-10-24, 2025.

[2] Atlassian, *Scrum*, https://www.atlassian.com/agile/scrum, Accessed: 2025-10-24, 2025.

[3] BJIT Group, *About bjit — outsourcing of software development in bangladesh*, https://bjitgroup.com/about-us, Accessed: August 10, 2025, 2024.

[4] BJIT Group, *Home — best offshore software development company*, https://bjitgroup.com/, Accessed: 2025-10-24, 2025.

[5] Amazon Web Services, *What is the software development lifecycle?* https://aws.amazon.com/what-is/sdlc/, Accessed: August 10, 2025, 2024.

[6] K. Schwaber and J. Sutherland, *The scrum guide*, https://scrumguides.org/scrum-guide.html, Accessed: August 10, 2025, 2020.

[7] BrowserStack, *Manual testing vs automation testing: Key differences*, https://www.browserstack.com/guide/manual-vs-automated-testing-differences, Accessed: August 11, 2025, 2024.

[8] N. Yadav, *Automation vs manual: A hybrid approach for effective qa*, https://www.testriq.com/blog/post/automation-vs-manual-a-hybrid-approach-for-effective-qa, Accessed: August 11, 2025, 2024.

Appendix A

Appendices

A.1 Daily/Weekly Training Schedule

The following table outlines the schedule followed during the two-week industrial attachment program at BJIT Ltd.

Table A.1: Industrial Attachment Activity Plan

Day	Date	Activities	Time
1	03-Mar-2025	Team Introduction, Scope and Schedule	10:00 AM
		Introduction to BJIT	2:00 PM
2	04-Mar-2025	Exposure to Industry Practices (SDLC, Tools)	10:00 AM
		Practical Experience (SRS, System Design)	2:00 PM
3	05-Mar-2025	Development Work (Environment Setup)	9:00 AM
		Collaboration (Scrum Team Practice)	10:00 AM
		Intro to Software Domains & Tech Trends	11:00 AM
		Real exposure with industry people	2:00 PM
4	06-Mar-2025	Project Development Work	Full Day
		Soft Skill (Communication)	10:00 AM
		SQA (Manual) - Hear from Industry Leaders	2:00 PM
5	07-Mar-2025	Continue Development Work	9:00 AM - 3:00 PM
		Soft Skill (High Performing Team Workplace Ethics)	10:00 AM
		SQA Automation - Hear from Industry Expert	2:00 PM

Table A.1 – continued from previous page

Day	Date	Activities	Time
6	10-Mar-2025	Continue Development Work Technology Introduction and Trend Self-directed Development Work	8:00 AM 10:00 AM 12:00 PM - 3:00 PM
7	11-Mar-2025	AI - Hear from Industry Leaders Follow-up Development Work	11:00 AM 12:00 PM
8	12-Mar-2025	Development Work Task Submission	8:00 AM 10:00 AM
9	13-Mar-2025	Demo of Development Work	11:00 AM - 3:00 PM
10	14-Mar-2025	Networking, Visit to Engineering Division, Q&A Closing & Photo Session	10:00 AM - 12:00 PM 2:00 PM - 3:00 PM

A.2 Main Topics of BJIT Academy Program

Table A.2: Main Topics and Session Schedule

Session	Topic
1	Exposure to Industry Practices
2	Collaboration
3	Practical Experience (Real Project Practice)
4	Hear from Industry Experts
5	Networking Opportunities
6	Soft Skills
7	Closing & Photo Session

A.3 Toolchain Used During Attachment

Table A.3: Tools and Platforms Used in the Project

Category	**Tool/Platform**	**Purpose in Attachment**
IDE/Editor	VS Code, PyCharm	Daily coding, debugging
VCS	Git (GitHub/GitLab)	Branching, PRs, code review
Issue Mgmt	Jira (or equivalent)	User stories, sprints, bugs
CI/CD	GitLab CI / GitHub Actions	Build–test–deploy pipeline
DB	PostgreSQL (dev)	Local testing & analytics
QA	pytest/JUnit, Postman	Unit/API testing

A.4 Skills Growth (self-assessment)

Table A.4: Self-evaluation before and after.

Skill Area	**Before (1–5)**	**After (1–5)**	**Evidence**
Python (OOP, file I/O)	3	4	Library/Task-manager code
Git (branches/PRs)	2	4	Feature branches merged
Agile (Scrum events)	2	4	Daily stand-ups; sprint review
CI/CD basics	1	3	Pipeline dry-run on staging