SOLFLOW

A COURSE PROJECT REPORT

submitted by

JOYAL JAISON (MUT24CA045)

NOEL PAUL TOMY (MUT24CA055)

ABEL SUNIL (MUT24CA03)

SAI NATH SANTHOSH (MUT24CA062)

as part of the course

PBCST304 OBJECT ORIENTED PROGRAMMING

to

The APJ Abdul Kalam Technological University in partial fullfilment of the requirements for the award of the Degree

of

Bachelor of Technology

In

Computer Science & Engineering

Department of Artificial Intelligence and Data Science

Muthoot Institute of Technology and Science

Kochi - 682308

OCTOBER 2025

CERTIFICATE

This is to certify that the report entitled **"SOLFLOW"**, *submitted by* **Joyal Jaison,Noel Paul Tomy,Abel Sunil,Sai Nath Santhosh** *to Muthoot Institute of Technology and Science, Kochi for the award of the degree of Bachelor of Technology in Computer Science & Engineering(Artificial Intelligence), is a bonafide record of the course project work carried out by them, as part of the course PBCST304 Object Oriented Programming, under my supervision and guidance.*

Ms. Nimmi M.K
Faculty-in-charge

Place:Kochi
Date:23/10/2025

ABSTRACT

This project presents Sol Flow, a Java-based business productivity system designed for solopreneurs seeking an all-in-one workspace to manage their daily operations efficiently. Modeled after Notion but focused solely on workflow optimization, the platform integrates essential modules such as a Daily Planner, Custom Spreadsheet, Mail Organizer, Document Upload Center, Client Manager, and Workflow Marketplace.

The system enables users to organize, plan, and execute their business activities within a unified interface—reducing distractions and eliminating the need to switch between multiple tools. Built using Object-Oriented Programming (OOP) principles, Sol Flow ensures modularity, scalability, and maintainability. It also includes features like reminders, notifications, and a credit-based storage upgrade system to enhance productivity and flexibility.

By combining intuitive design with practical utility, Sol Flow empowers solopreneurs to manage tasks, track progress, and streamline workflows efficiently, fostering focus, consistency, and growth.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PURPOSE OF THE SYSTEM

The Sol Flow System is designed as a comprehensive digital workspace tailored for solopreneurs who manage multiple aspects of their business independently. The purpose of this system is to unify daily planning, client management, document handling, and communication under one platform. Implemented using object-oriented programming principles, the system emphasizes modularity and scalability, ensuring that each feature—such as task management, spreadsheets, and reminders—operates as an independent class interacting through well-defined interfaces. This reduces redundancy, simplifies maintenance, and allows for future expansion without architectural changes.

1.2 RELEVANCE AND SCOPE OF THE SYSTEM

The Sol Flow system is highly relevant in the current era of remote work and digital entrepreneurship. Solopreneurs frequently rely on multiple applications—Google Calendar, Excel, Notion, and email clients—to manage tasks, communications, and data. This fragmented approach leads to inefficiency and mental fatigue. Sol Flow solves this by offering a unified workspace specifically built for individuals running small businesses, startups, or personal brands.

The system's scope extends from daily task management to strategic business organization. Its modular architecture allows for easy integration of future enhancements such as real-time collaboration, cloud-based data storage, AI-powered scheduling assistants, and workflow analytics. Through this project, key object-oriented concepts are practically applied to create a scalable, secure, and user-friendly solution that aligns with real-world business productivity needs.

CHAPTER 2

PROBLEM DEFINITION

2.1 PROBLEM STATEMENT

In today's fast-paced digital environment, solopreneurs are required to handle multiple roles simultaneously—managing clients, planning tasks, organizing data, and maintaining communication. Most rely on separate tools for each of these purposes, which leads to frequent context switching, inefficiency, and distraction. The lack of an integrated platform results in lost productivity and time. There is a strong need for a focused, distraction-free, and unified productivity system that allows solopreneurs to plan, execute, and manage their workflows efficiently from a single workspace. Sol Flow aims to fill this gap by providing an all-in-one productivity environment designed for workflow efficiency and focus.

2.2 OBJECTIVES

The main objectives of **Sol Flow** are:

- To develop a centralized productivity system that integrates essential business management functions into a single cohesive platform.
- To enhance workflow efficiency by minimizing tool-switching and improving focus for solopreneurs.
- To design the system using Object-Oriented Programming principles for scalability, modularity, and maintainability.
- To provide a clean, user-friendly interface with intuitive drag-and-drop interactions.
- To implement a credit-based storage model that allows flexible upgrades as user needs grow.
- To build a Workflow Marketplace where users can create, share, or sell custom productivity templates.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

The functional requirements define the key capabilities that the Sol Flow system must provide to its users. These features form the foundation of the application's functionality and ensure that it supports essential solopreneur workflows effectively.

- **User Authentication and Profile Management:** Allow users to sign up, log in, and manage their profiles securely.
- **Daily Planner:** Add, edit, delete, and prioritize tasks with a time-blocking calendar view.
- **Custom Spreadsheet Module:** Create and edit lightweight spreadsheets for tracking and analysis.
- **Mail Organizer:** Compose, categorize, and format professional emails with tagging support.
- **Document Upload Center:** Upload, preview, and organize PDF and document files while tracking storage usage.
- **Client/Contact Manager:** Store client information, filter and search contacts, and track client status.
- **Notification System:** Send reminders for missed or pending tasks and generate daily summaries.
- **Workflow Marketplace:** Enable users to create, share, and purchase workflow templates.

3.2 NON-FUNCTIONAL REQUIREMENTS

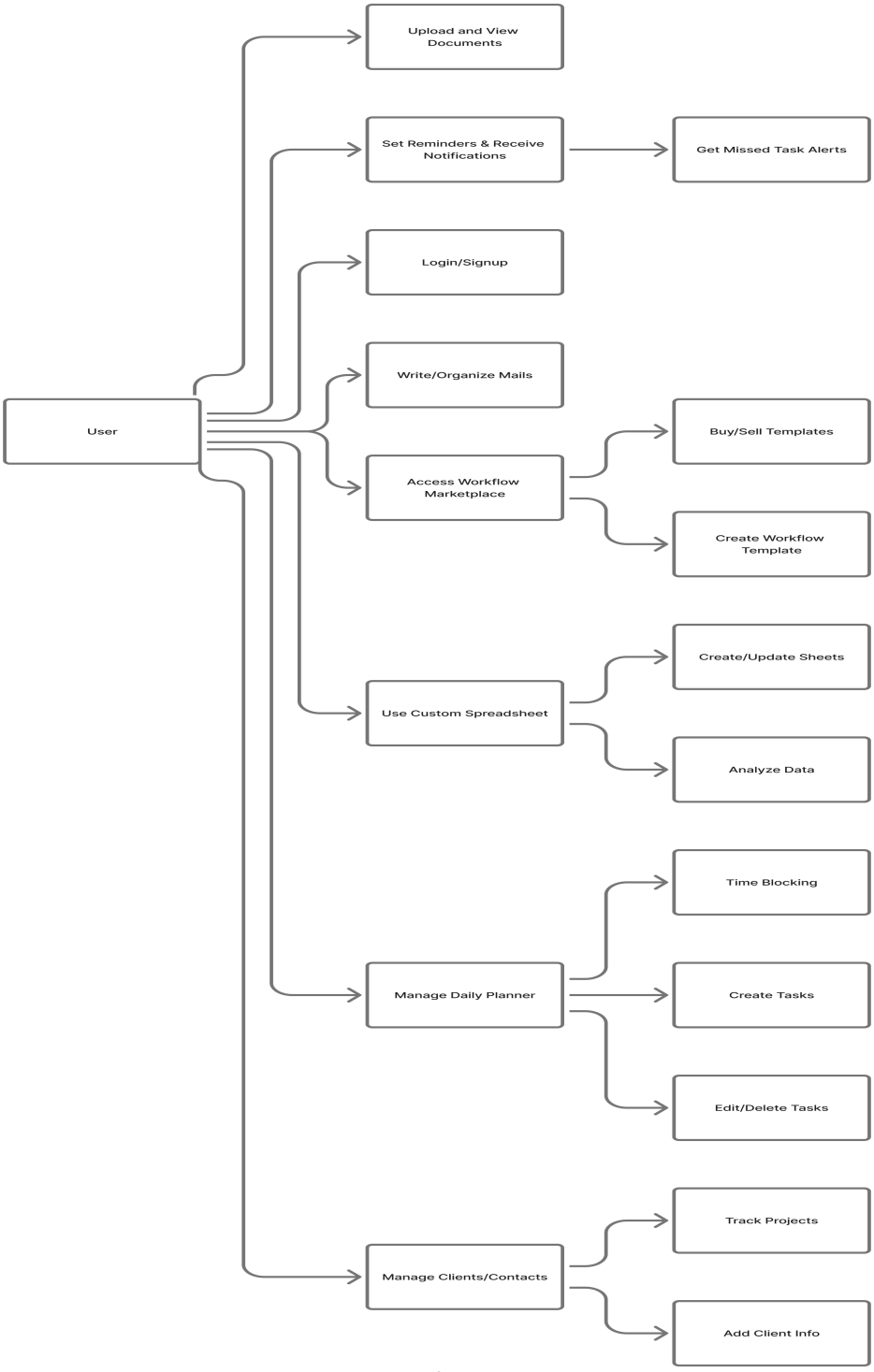
The non-functional requirements define the performance, security, and design characteristics that determine how well the system will operate. These aspects ensure that Sol Flow remains efficient, secure, and user-friendly under all conditions.

- **Performance:** The system should perform efficiently with minimal latency.
- **Scalability:** The architecture must support easy feature addition and storage expansion.
- **Security:** Ensure user data and documents are securely stored and encrypted.
- **Usability:** The interface should be intuitive, minimalistic, and distraction-free.
- **Reliability:** Maintain data integrity and provide regular backups to prevent loss.
- **Portability:** The web-based design should be browser-compatible and adaptable for mobile use.
- **Maintainability:** Code should follow OOP standards for easy updates, debugging, and scaling.

CHAPTER 4

SYSTEM DESIGN

4.1 USE CASE DIAGRAM



4.2 CLASS DIAGRAM

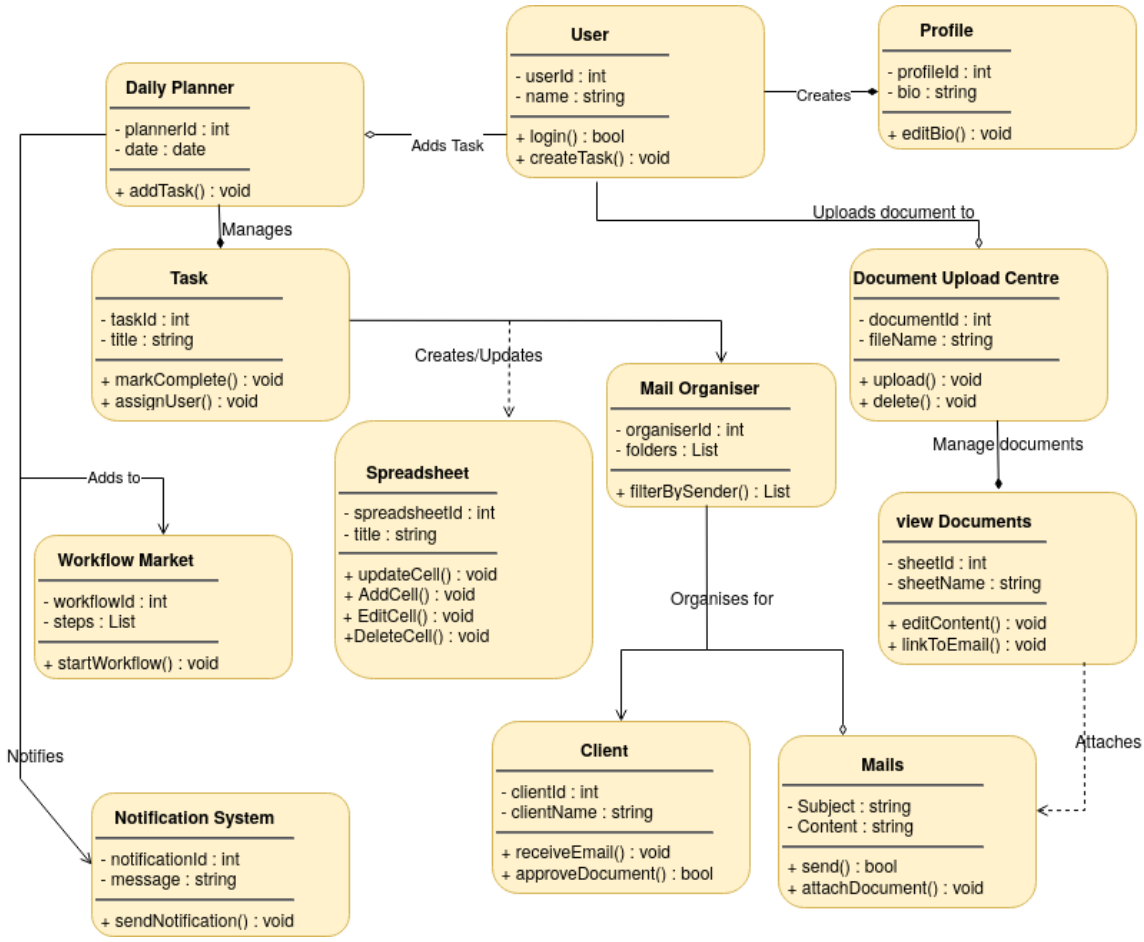


Figure 4.1: Class Diagram

CHAPTER 5

IMPLEMENTATION

5.1 CODE SNIPPETS

```
package main.ui;

import javax.swing.*;
import java.awt.*;

public class NavigationBar extends JPanel {
    private final JButton backBtn;
    private final JButton forwardBtn;

    public NavigationBar() {
        setLayout(new BorderLayout());
        setBackground(new Color(24, 26, 28));
        setBorder(BorderFactory.createEmptyBorder(8, 12, 8, 12));

        JPanel left = new JPanel(new FlowLayout(FlowLayout.LEFT, 8, 0));
        left.setOpaque(false);

        ImageIcon backIcon = null;
        ImageIcon forwardIcon = null;
        java.net.URL backUrl = getClass().getResource("/assets/back.png");
        java.net.URL forwardUrl = getClass().getResource("/assets/forward.png");
        if (backUrl != null) backIcon = new ImageIcon(new ImageIcon(backUrl).getImage().getScaledInstance(16, 16, Image.SCALE_SMOOTH));
        if (forwardUrl != null) forwardIcon = new ImageIcon(new ImageIcon(forwardUrl).getImage().getScaledInstance(16, 16, Image.SCALE_SMOOTH));

        backBtn = (backIcon != null) ? new JButton(backIcon) : new JButton("←");
        backBtn.setToolTipText("Back");
        backBtn.setFocusPainted(false);
        backBtn.setForeground(Color.WHITE);
        backBtn.setBackground(new Color(48, 51, 57));
        backBtn.setBorder(BorderFactory.createEmptyBorder(6,10,6,10));
        backBtn.addActionListener(e -> { try { NotionStyleUI.navigateToHome(); } catch (Exception ignored) {} });

        forwardBtn = (forwardIcon != null) ? new JButton(forwardIcon) : new JButton("→");
        forwardBtn.setToolTipText("Forward");
        forwardBtn.setFocusPainted(false);
        forwardBtn.setForeground(Color.WHITE);
        forwardBtn.setBackground(new Color(48, 51, 57));
        forwardBtn.setBorder(BorderFactory.createEmptyBorder(6,10,6,10));
        forwardBtn.setEnabled(false);

        UITheme.styleNavButton(backBtn);
        UITheme.styleNavButton(forwardBtn);
        left.add(backBtn);
        left.add(forwardBtn);

        JLabel title = new JLabel("SolFlow");
        title.setForeground(Color.WHITE);
        title.setFont(new Font("SansSerif", Font.BOLD, 14));

        add(left, BorderLayout.WEST);
        add(title, BorderLayout.CENTER);
    }

    public static NavigationBar create() { return new NavigationBar(); }
    public static JPanel wrap(JComponent page) {
        JPanel wrapper = new JPanel(new BorderLayout());
        wrapper.setOpaque(false);
        wrapper.add(create(), BorderLayout.NORTH);
        wrapper.add(page, BorderLayout.CENTER);
        return wrapper;
    }

    public JButton getBackButton() { return backBtn; }
    public JButton getForwardButton() { return forwardBtn; }
}
```

Figure 5.1: Source Code for Navigation bar

```
private static void showLoginPanel() {
    mainContent.removeAll();
    mainContent.setLayout(new BorderLayout());

    JPanel center = new JPanel(new GridBagLayout());
    center.setOpaque(false);

    JPanel card = new JPanel() {
        @Override protected void paintComponent(Graphics g) {
            Graphics2D g2 = (Graphics2D) g.create();
            g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
            g2.setColor(new Color(30, 32, 36));
            g2.fillRoundRect(0, 0, getWidth(), getHeight(), 18, 18);
            g2.dispose();
            super.paintComponent(g);
        }
    };
    card.setOpaque(false);
    card.setLayout(new GridBagLayout());
    card.setBorder(BorderFactory.createEmptyBorder(24, 28, 24, 28));

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(8, 8, 8, 8);
    gbc.fill = GridBagConstraints.HORIZONTAL;

    JLabel title = new JLabel("Welcome to SolFlow");
    title.setForeground(Color.WHITE);
    title.setFont(new Font("SansSerif", Font.BOLD, 22));
    gbc.gridx = 0; gbc.gridy = 0; gbc.gridwidth = 2; card.add(title, gbc);

    gbc.gridwidth = 1;
    gbc.gridy = 2; gbc.gridx = 0;
    JLabel uLabel = new JLabel("Username"); uLabel.setForeground(new Color(200,200,200)); uLabel.setFont(new Font("SansSerif", Font.PLAIN, 12));
    card.add(uLabel, gbc);
    gbc.gridx = 1;
    JTextField userField = new JTextField(18);
    userField.setBackground(new Color(40,40,40)); userField.setForeground(Color.WHITE); userField.setBorder(BorderFactory.createEmptyBorder(8,8,8,8));
    card.add(userField, gbc);

    gbc.gridy = 3; gbc.gridx = 0;
    JLabel pLabel = new JLabel("Password"); pLabel.setForeground(new Color(200,200,200)); pLabel.setFont(new Font("SansSerif", Font.PLAIN, 12));
    card.add(pLabel, gbc);
    gbc.gridx = 1;
    JPasswordField passField = new JPasswordField(18);
    passField.setBackground(new Color(40,40,40)); passField.setForeground(Color.WHITE); passField.setBorder(BorderFactory.createEmptyBorder(8,8,8,8));
    card.add(passField, gbc);

    gbc.gridy = 5; gbc.gridwidth = 1; gbc.gridx = 0;
    JButton loginBtn = new JButton("Sign in");
    loginBtn.setBackground(new Color(88,101,242)); loginBtn.setForeground(Color.WHITE); loginBtn.setFocusPainted(false);
    card.add(loginBtn, gbc);

    gbc.gridx = 1;
    JButton registerBtn = new JButton("Create account");
    registerBtn.setBackground(new Color(58,58,58)); registerBtn.setForeground(Color.WHITE); registerBtn.setFocusPainted(false);
    card.add(registerBtn, gbc);

    JLabel msg = new JLabel(" "); msg.setForeground(new Color(220,100,100));
    gbc.gridy = 4; gbc.gridx = 0; gbc.gridwidth = 2; card.add(msg, gbc);

    center.add(card);
    mainContent.add(center, BorderLayout.CENTER);

    loginBtn.addActionListener(e -> {
        String u = userField.getText().trim();
        String p = new String(passField.getPassword());
        if (u.isEmpty() || p.isEmpty()) { msg.setText("Please enter username and password"); return; }
        try {
            main.db.UserDAO.AuthRecord ar = main.db.UserDAO.getAuthByUsername(u);
            if (ar == null) { msg.setText("User not found. Please register."); return; }
            String hashed = HashUtil.hashWithSalt(p, ar.salt);
            if (hashed.equals(ar.passwordHash)) { main.Session.set(ar.id, ar.username); showHomePage(); }
            else { msg.setText("Invalid credentials"); }
        } catch (Exception ex) { ex.printStackTrace(); msg.setText("Error during login"); }
    });

    registerBtn.addActionListener(e -> {
        String u = userField.getText().trim();
        String p = new String(passField.getPassword());
        if (u.isEmpty() || p.isEmpty()) { msg.setText("Please enter username and password"); return; }
        try {
            String salt = HashUtil.generateSalt();
            String hash = HashUtil.hashWithSalt(p, salt);
            int id = main.db.UserDAO.insert(u, hash, salt);
            if (id > 0) { main.Session.set(id, u); showHomePage(); } else { msg.setText("Registration failed (username may exist)"); }
        } catch (Exception ex) { ex.printStackTrace(); msg.setText("Error during registration"); }
    });

    mainContent.revalidate(); mainContent.repaint();
}
```

Figure 5.2: Source Code for login module

```
public MailOrganize() {
    setLayout(new BorderLayout());
    setBackground(APP_BG);
    setBorder(new EmptyBorder(12,12,12,12));

    JPanel pageContent = new JPanel(new BorderLayout());
    pageContent.setOpaque(false);

    JPanel top = new JPanel(new BorderLayout(12, 12));
    top.setBackground(APP_BG);
    top.setBorder(new EmptyBorder(6, 6, 6, 6));

    JLabel title = new JLabel("Sheets");
    title.setFont(HEAD_FONT);
    title.setForeground(new Color(26,26,26));
    top.add(title, BorderLayout.WEST);

    JButton importBtn = createToolbarBtn("Import CSV", "/assets/csv-file-format-extension.png", new Color(64,160,255));
    pageContent.add(top, BorderLayout.NORTH);

    File projectRoot = new File(System.getProperty("user.dir"));
    rootNode = new DefaultMutableTreeNode(new FileNode(projectRoot));
    tree = new JTree(rootNode);
    populateChildren(rootNode);

    JScrollPane leftScroll = new JScrollPane(tree);
    leftScroll.setPreferredSize(new Dimension(320, 640));

    JPanel right = new JPanel(new BorderLayout(10,10));
    right.setBackground(APP_BG);

    JPanel card = new JPanel(new BorderLayout(8,8));
    card.setBackground(SURFACE);

    table.setFillsViewportHeight(true);
    table.setFont(TABLE_FONT);
    table.setRowHeight(34);

    JScrollPane tableScroll = new JScrollPane(table);
    tableScroll.setBorder(BorderFactory.createEmptyBorder());
    card.add(tableScroll, BorderLayout.CENTER);

    right.add(card, BorderLayout.CENTER);
    JSplitPane split = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, leftScroll, right);
    split.setResizeWeight(0.28);
    pageContent.add(split, BorderLayout.CENTER);

    add(NavigationBar.wrap(pageContent), BorderLayout.CENTER);
}

private void loadCsvFile(File f) {
    if (f == null || !f.exists()) return;
    try (BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(f), StandardCharsets.UTF_8))) {
        String line;
        List<String[]> rows = new ArrayList<>();
        while ((line = br.readLine()) != null) rows.add(parseCsvLine(line));
        if (rows.isEmpty()) return;
        int maxCols = 0;
        for (String[] r : rows) if (r.length > maxCols) maxCols = r.length;

        tableModel.clear();
        for (int c = 0; c < maxCols; c++) tableModel.addColumn("C" + (c+1));
        for (String[] r : rows) {
            String[] row = new String[maxCols];
            for (int i = 0; i < maxCols; i++) row[i] = (i < r.length) ? r[i] : "";
            tableModel.addRow(row);
        }

        try { FileDAO.insertOrUpdate(f); } catch (Exception ignored) {}
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(this, "Failed to load CSV: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

Figure 5.3: Source Code for Mail Organizer module

```
public static JPanel createEmbeddedPanel(int workflowId) {
    InbuiltJavaTemplate1 t = new InbuiltJavaTemplate1(workflowId);
    return NavigationBar.wrap(t.getPanel());
}

private void loadTasksForWorkflow() {
    tableModel.setRowCount(0);
    List<TaskDAO.TaskRecord> tasks = TaskDAO.listForWorkflow(this.workflowId <= 0 ? 0 : this.workflowId);
    for (TaskDAO.TaskRecord t : tasks) {
        tableModel.addRow(new Object[] { t.text, t.checked });
    }
}

private void createNewTask(String text) {
    if (text == null || text.trim().isEmpty()) return;
    int ord = tableModel.getRowCount();
    int id = TaskDAO.insert(text, false, ord, this.workflowId <= 0 ? 0 : this.workflowId);
    if (id != -1) loadTasksForWorkflow();
    else tableModel.addRow(new Object[] { text, false });
}
```

Figure 5.4: Source Code for Embedded Panels

```
public static JPanel createEmbeddedPanel() {
    return new NewTemplatePanel();
}

private static class BreadcrumbPanel extends JPanel {
    private final JLabel activeModuleLabel = new JLabel("Sales");
    private final JPopupMenu navMenu = new JPopupMenu();
    private final java.util.function.Consumer<String> onNavigate;

    BreadcrumbPanel(java.util.function.Consumer<String> onNavigate) {
        super(new BorderLayout(10, 10));
        this.onNavigate = onNavigate;
        setBackground(Theme.BACKGROUND);
        setBorder(new EmptyBorder(15, 20, 10, 20));

        navMenu.add(createNavItem("Opportunities", "JOB_SEARCH"));
        navMenu.add(createNavItem("Clients", "CAREER_GOALS"));
        navMenu.add(createNavItem("Activity", "ACTION_ITEMS"));

        JPanel breadcrumbPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 5, 0));
        breadcrumbPanel.setOpaque(false);
        breadcrumbPanel.add(new IconOnlyButton(IconType.BRIEFCASE));

        JButton workButton = new FlatButton("Sales & Clients");
        workButton.setFont(Theme.MUTED_FONT);
        workButton.addActionListener(e -> navMenu.show(workButton, 0, workButton.getHeight
        breadcrumbPanel.add(workButton);

        JLabel separatorLabel = new JLabel("/");
        separatorLabel.setFont(Theme.MUTED_FONT);
        separatorLabel.setForeground(Theme.TEXT_MUTED);
        breadcrumbPanel.add(separatorLabel);

        activeModuleLabel.setFont(Theme.MUTED_FONT.deriveFont(Font.BOLD));
        activeModuleLabel.setForeground(Theme.TEXT);
        breadcrumbPanel.add(activeModuleLabel);

        add(breadcrumbPanel, BorderLayout.WEST);
    }

    private JMenuItem createNavItem(String text, String moduleName) {
        JMenuItem item = new JMenuItem(text);
        item.setFont(Theme.BODY_FONT);
        item.addActionListener(e -> onNavigate.accept(moduleName));
        return item;
    }

    void setActiveModule(String moduleName) {
        switch(moduleName) {
            case "JOB_SEARCH": activeModuleLabel.setText("Opportunities"); break;
            case "CAREER_GOALS": activeModuleLabel.setText("Clients"); break;
            case "ACTION_ITEMS": activeModuleLabel.setText("Activity"); break;
        }
    }
}
```

Figure 5.5: Source Code for Embedded Panels Implementation

```
package main;

import javax.swing.SwingUtilities;
import main.ui.NotionStyleUI;
import main.db.Database;

public class Main {
    public static void main(String[] args) {
        Database.init();
        SwingUtilities.invokeLater(() -> NotionStyleUI.show());
    }
}
```

`src/main/java/main/Session.java` (in-memory session holder)

```java
package main;

public final class Session {
    private static int currentUserId = 0;
    private static String currentUsername = null;

    public static void set(int userId, String username) { currentUserId = userId; currentU
    public static void clear() { currentUserId = 0; currentUsername = null; }
    public static int getUserId() { return currentUserId; }
    public static String getUsername() { return currentUsername; }
}
```

Figure 5.6: Source Code for User Session handler

```
package main.db;

import java.sql.*;
import java.io.File;

public final class Database {
    private static final String DB_NAME = "SolFlow.db";
    private static final String URL = "jdbc:sqlite:" + System.getProperty("user.dir") + File.separator + DB_NAME;

    public static Connection getConnection() throws SQLException {
        try { Class.forName("org.sqlite.JDBC"); } catch (ClassNotFoundException ignored) {}
        return DriverManager.getConnection(URL);
    }

    public static void init() {
        try (Connection c = getConnection(); Statement s = c.createStatement()) {
            s.execute("PRAGMA foreign_keys = ON;");
            s.execute("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT NOT NULL UNIQUE, password_hash TEXT NOT NULL, salt TEXT NOT NULL, created_at INTEGER)");

        } catch (SQLException ex) { ex.printStackTrace(); }
    }
}
```

Figure 5.7: Source Code for Database Connection

```
package main.ui;

import javax.swing.*;
import java.awt.*;
import main.db.TaskDAO;
import java.util.List;

public class InbuiltJavaTemplate1 {
    public static JPanel createEmbeddedPanel(int workflowId) {
        InbuiltJavaTemplate1 t = new InbuiltJavaTemplate1(workflowId);
        return NavigationBar.wrap(t.getPanel());
    }

    private JPanel getPanel() { return new JPanel(); }

    static class PlannerTasksPage extends JPanel {
        private final int workflowId;
        public PlannerTasksPage(Runnable backAction, int workflowId) {
            this.workflowId = workflowId;
            setLayout(new BorderLayout());
            loadTasksForWorkflow();
        }
        private void loadTasksForWorkflow() {
            List<TaskDAO.TaskRecord> tasks = TaskDAO.listForWorkflow(this.workflowId <= 0 ? 0 : this.workflowId);
        }
        private void createNewTask(String text) {
            if (text == null || text.trim().isEmpty()) return;
            int ord = 0;
            int id = TaskDAO.insert(text, false, ord, this.workflowId <= 0 ? 0 : this.workflowId);
            if (id != -1) loadTasksForWorkflow();
        }
    }

    static class DailyRoutinesPage extends JPanel {
        public DailyRoutinesPage(Runnable backAction) {
            setLayout(new BorderLayout());
            createRoutineList();
        }
        private void createRoutineList() {
            add(new JLabel("🧘 Meditate"));
            add(new JLabel("💪 Exercise"));
        }
    }
}
```

Figure 5.8: Source Code for Inbuilt Template 2

```
package main.ui;

import javax.swing.*;
import java.awt.*;

public class ContentCreatorApp extends JPanel {
    private final JLayeredPane canvas;
    private final JScrollPane canvasScroll;
    private JButton addBtn;
    private JButton googleBtn;

    public ContentCreatorApp() {
        setLayout(new BorderLayout());
        this.canvas = new JLayeredPane();
        this.canvasScroll = new JScrollPane(canvas);
        add(NavigationBar.wrap(new JPanel(new BorderLayout())), BorderLayout.CENTER);
    }

    private void addCard(String title, int x, int y, int w, int h) {
        DraggableCard card = new DraggableCard(title, x, y, w, h);
        canvas.add(card, JLayeredPane.DEFAULT_LAYER);
        canvas.revalidate();
        canvas.repaint();
    }

    private class DraggableCard extends JPanel {
        public DraggableCard(String title, int x, int y, int w, int h) {
            setLayout(new BorderLayout());
            setBounds(x, y, w, h);
            JButton removeButton = new JButton("✖");
            removeButton.addActionListener(e -> { canvas.remove(this); canvas.revalidate(); canvas.repaint(); });
            add(removeButton, BorderLayout.NORTH);
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame f = new JFrame("Customize the Contents (Preview)");
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            f.setSize(1200, 800);
            f.add(new ContentCreatorApp(), BorderLayout.CENTER);
            f.setVisible(true);
        });
    }
}
```

Figure 5.9: Source Code for Content Creation module

```java
package main.ui;

import javax.swing.*;
import java.awt.*;
import java.io.File;

public class Docupload extends JPanel {
    private final JPanel listPanel = new JPanel();
    private final java.util.List<File> uploadedFiles = new java.util.ArrayList<>();
    private final int workflowId;

    public Docupload() { this(0); }
    public Docupload(int workflowId) {
        this.workflowId = workflowId;
        setLayout(new BorderLayout());
        JPanel pageContent = new JPanel(new BorderLayout());
        JButton uploadBtn = createUploadButton("Upload", UITheme.ACCENT, UITheme.SURFACE);
        pageContent.add(uploadBtn, BorderLayout.NORTH);
        add(NavigationBar.wrap(pageContent), BorderLayout.CENTER);
        rebuildList();
    }

    private void rebuildList() {
        listPanel.removeAll();
        synchronized (uploadedFiles) {
            for (File f : uploadedFiles) listPanel.add(new JLabel(f.getName()));
        }
        listPanel.revalidate(); listPanel.repaint();
    }

    private JButton createUploadButton(String text, Color accent, Color surface) {
        JButton btn = new JButton(text);
        btn.addActionListener(e -> {
            JFileChooser fc = new JFileChooser();
            fc.setMultiSelectionEnabled(true);
            if (fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
                for (File f : fc.getSelectedFiles()) { if (!uploadedFiles.contains(f)) uploadedFiles.add(f); }
                rebuildList();
            }
        });
        return btn;
    }
}
```

Figure 5.10: Source Code for Document Uploader module

5.2 OUTPUT SCREENSHOTS

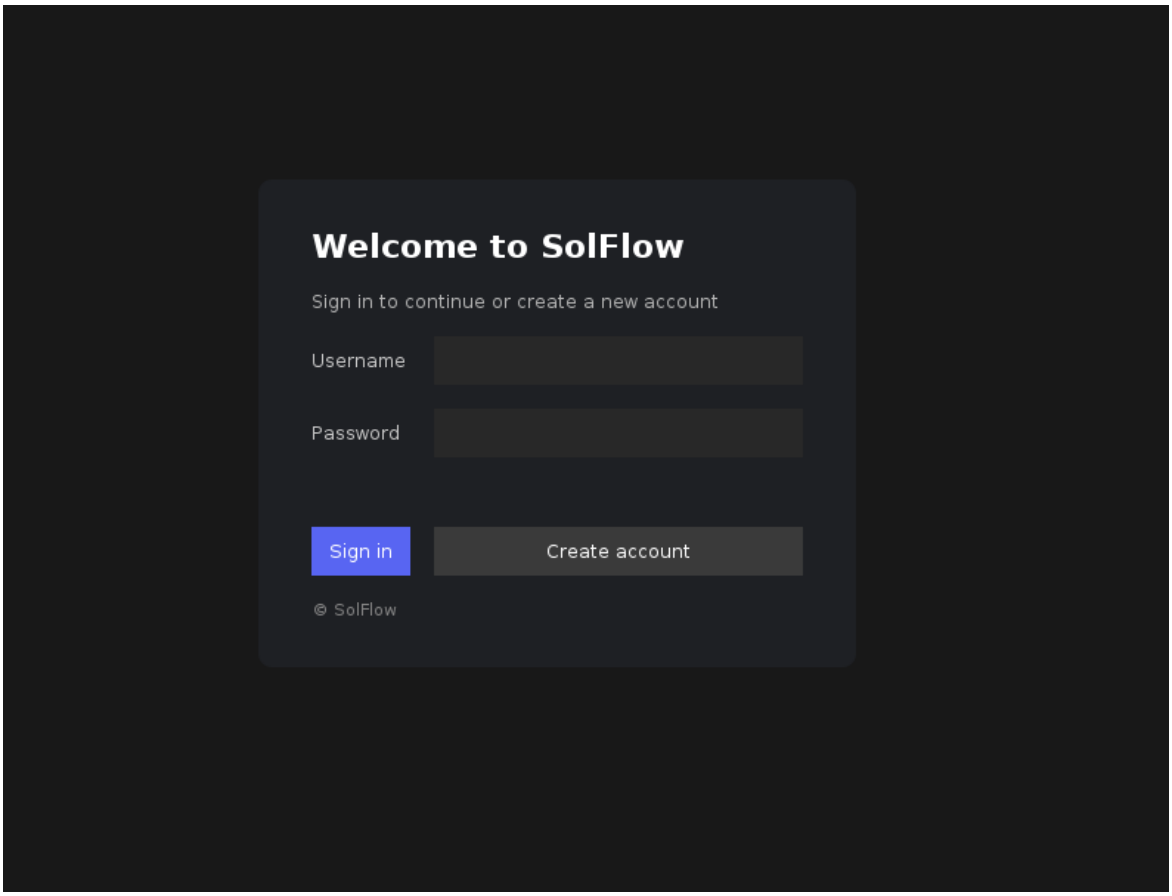


Figure 5.11: Account creation and Login

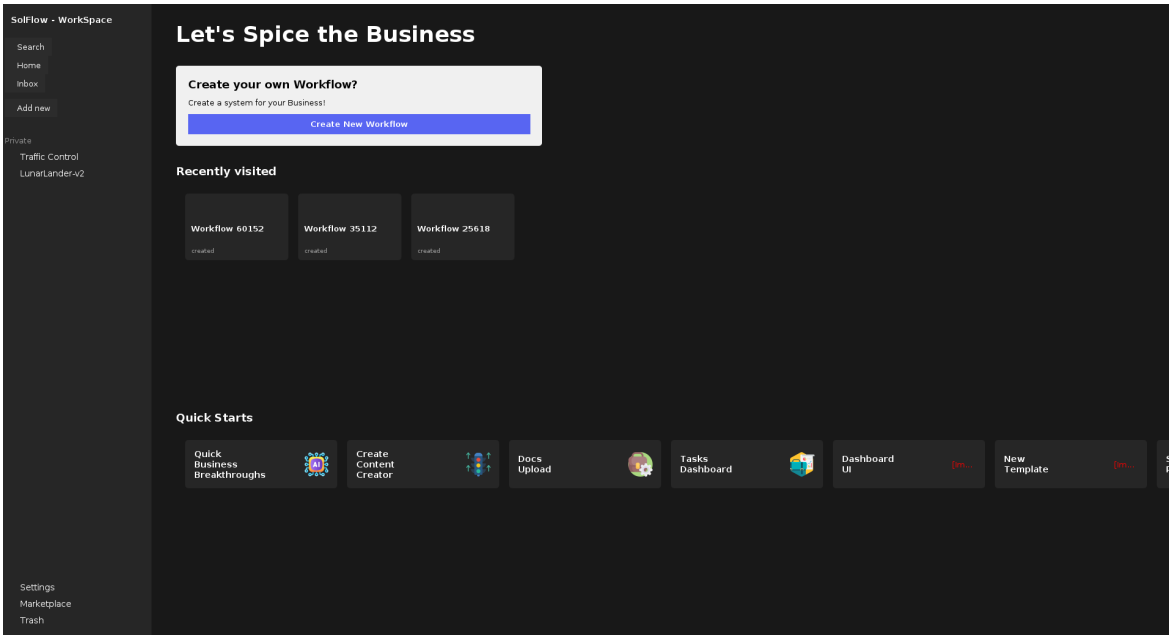


Figure 5.12: SolFlow Home

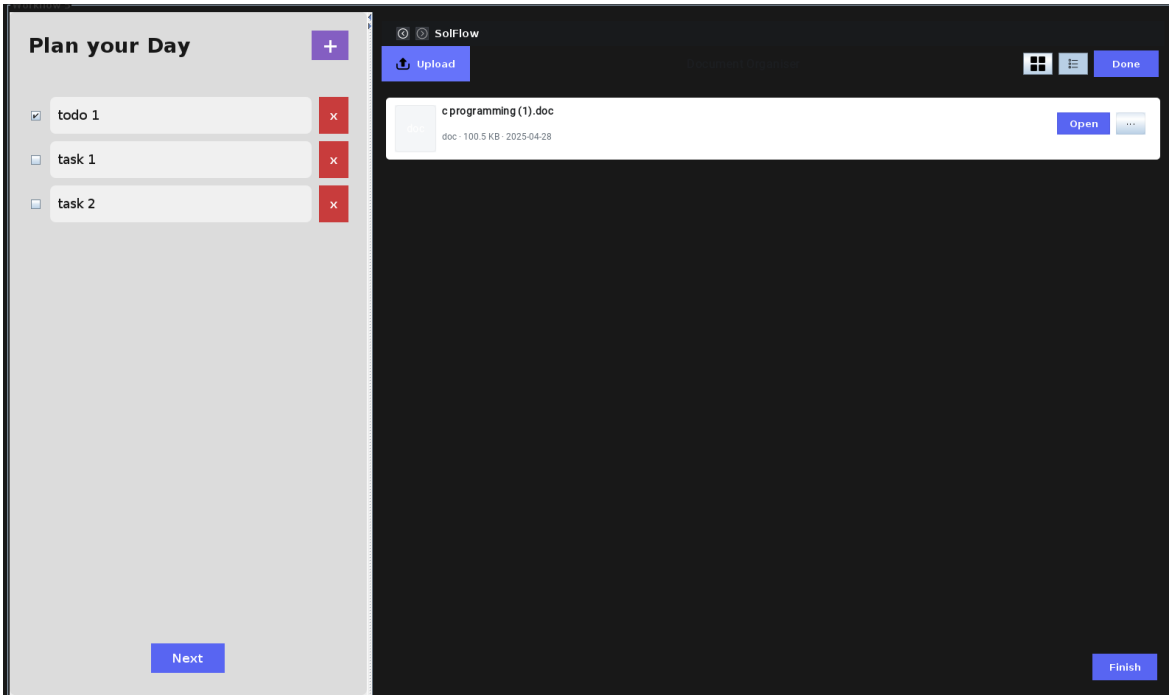


Figure 5.13: Document Uploader

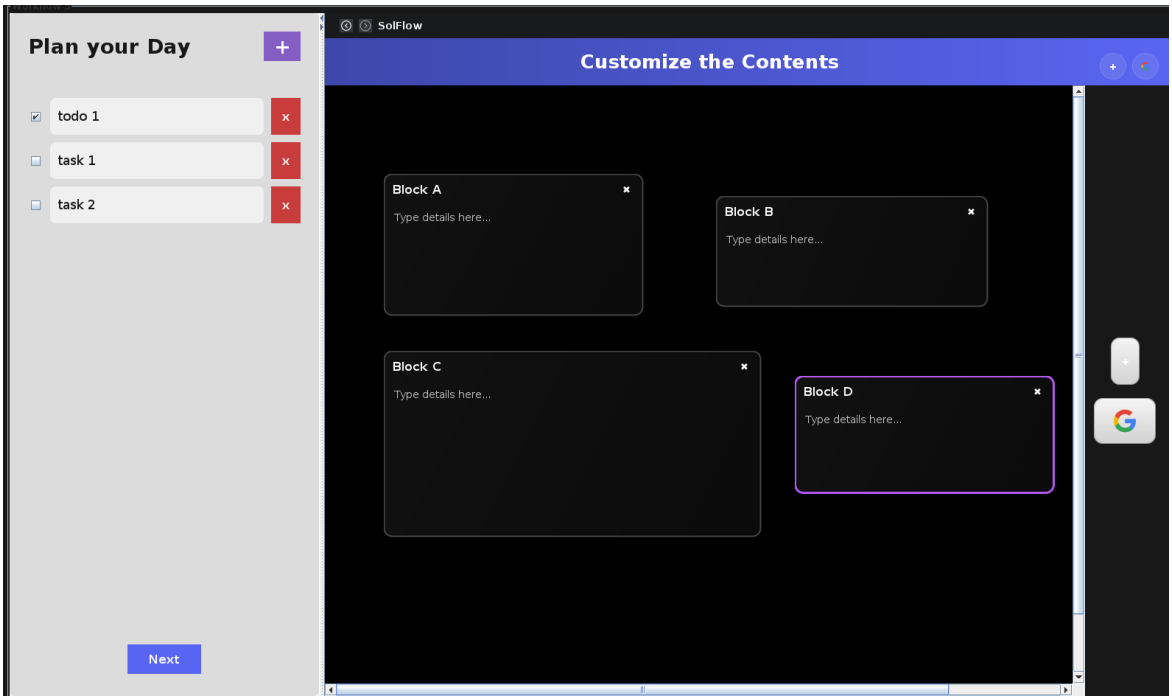


Figure 5.14: Content Creator

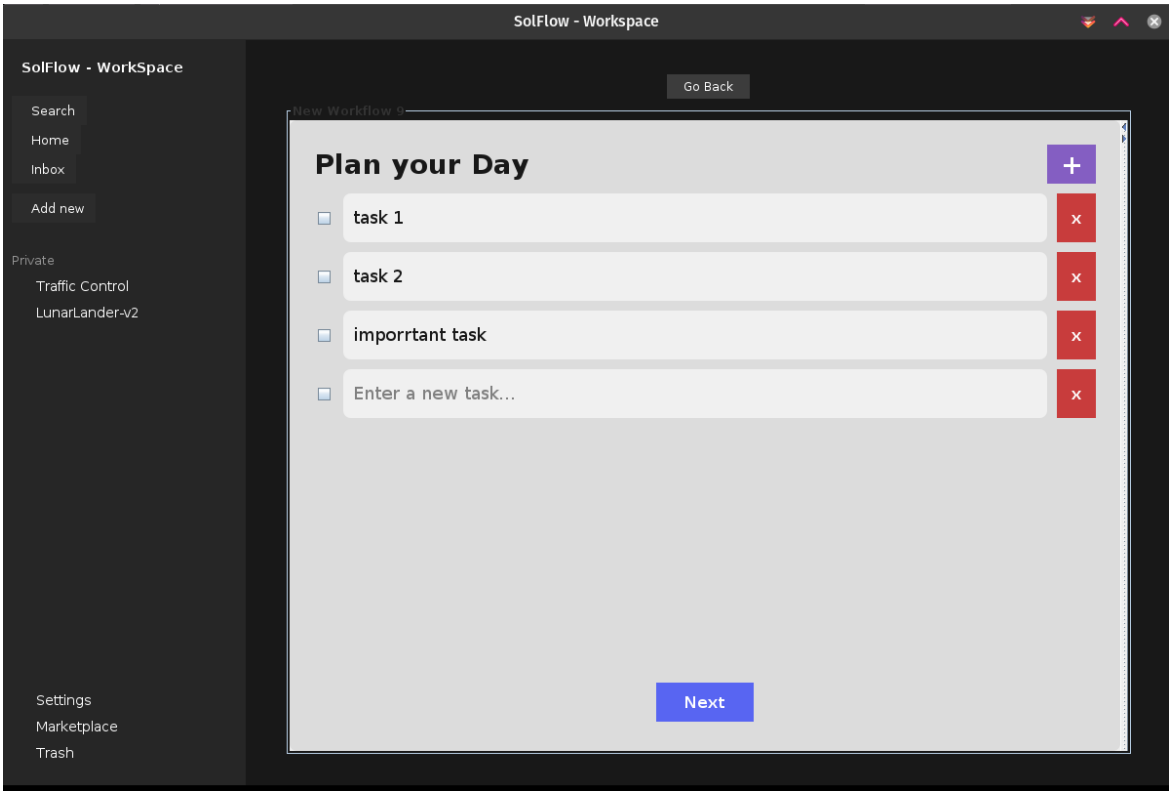


Figure 5.15: Daily Planner

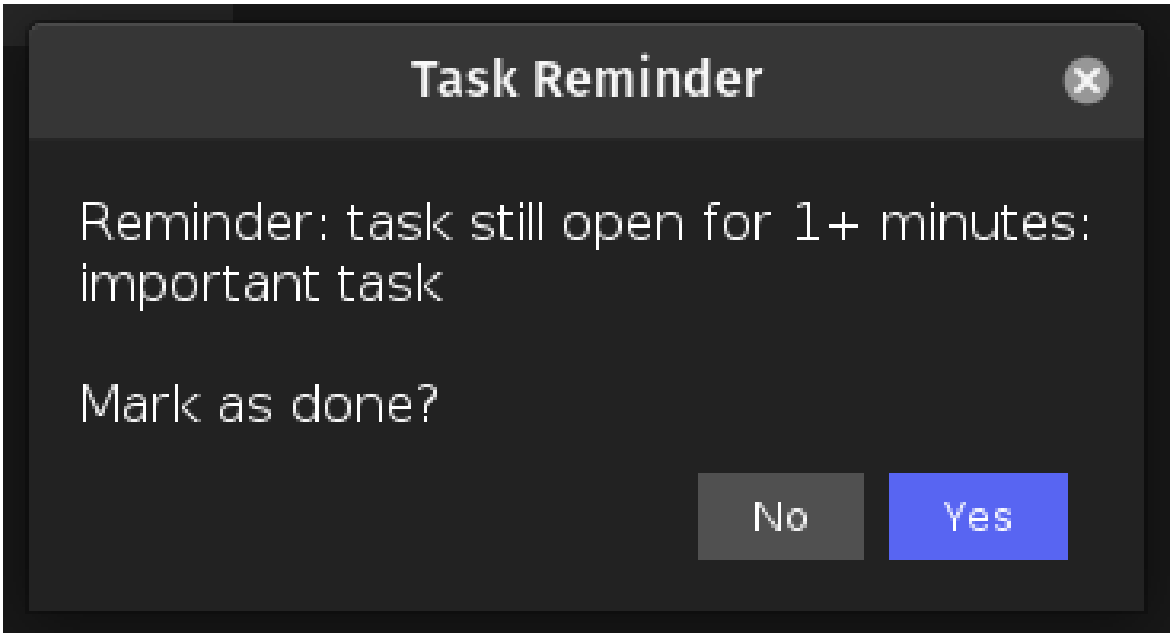


Figure 5.16: Task Reminder

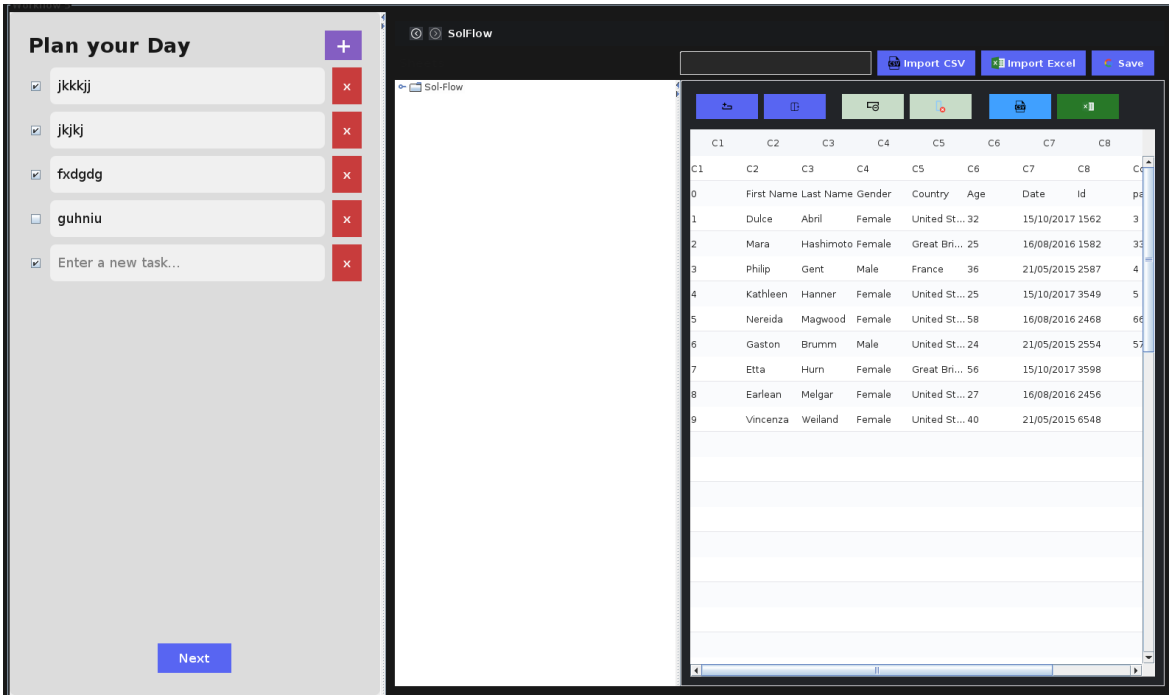


Figure 5.17: Mail Organizer

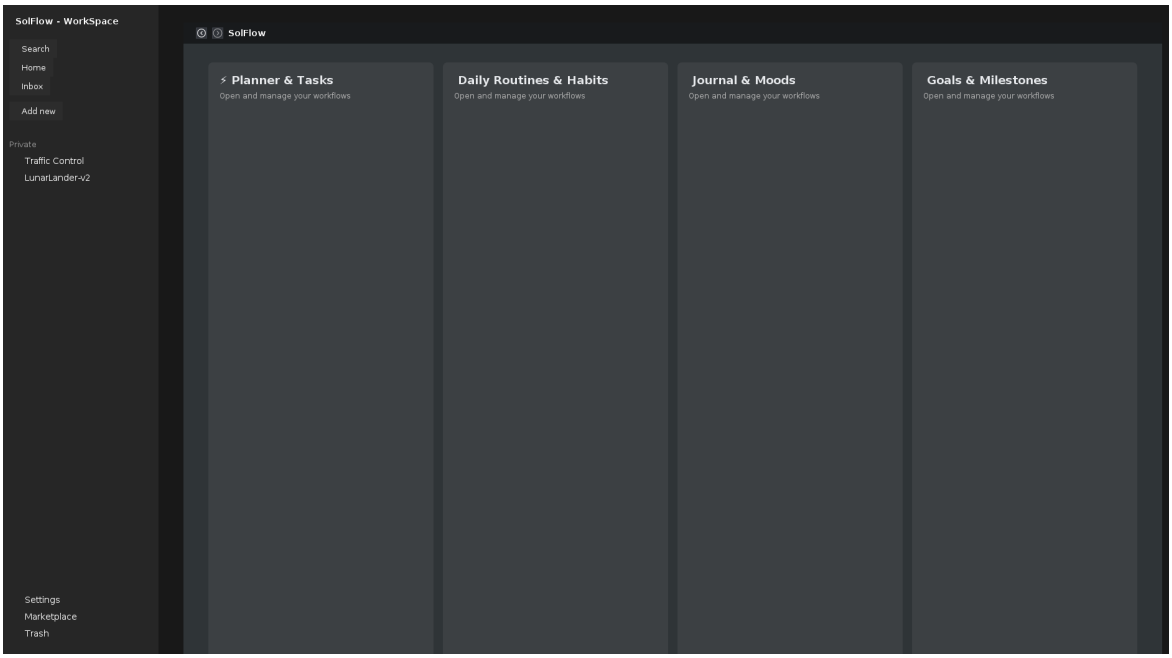


Figure 5.18: Inbuilt Template 1

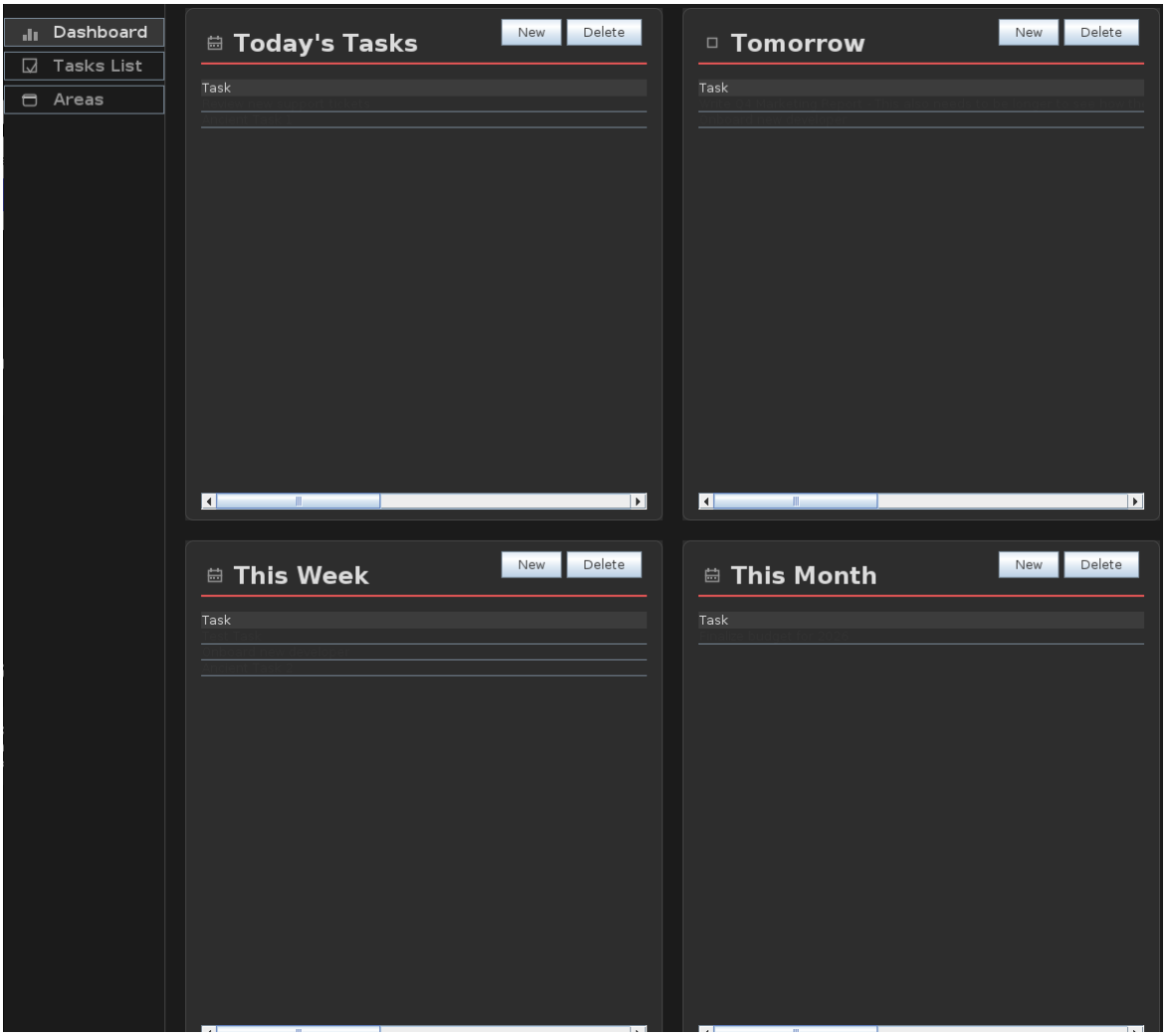


Figure 5.19: Inbuilt Template 2

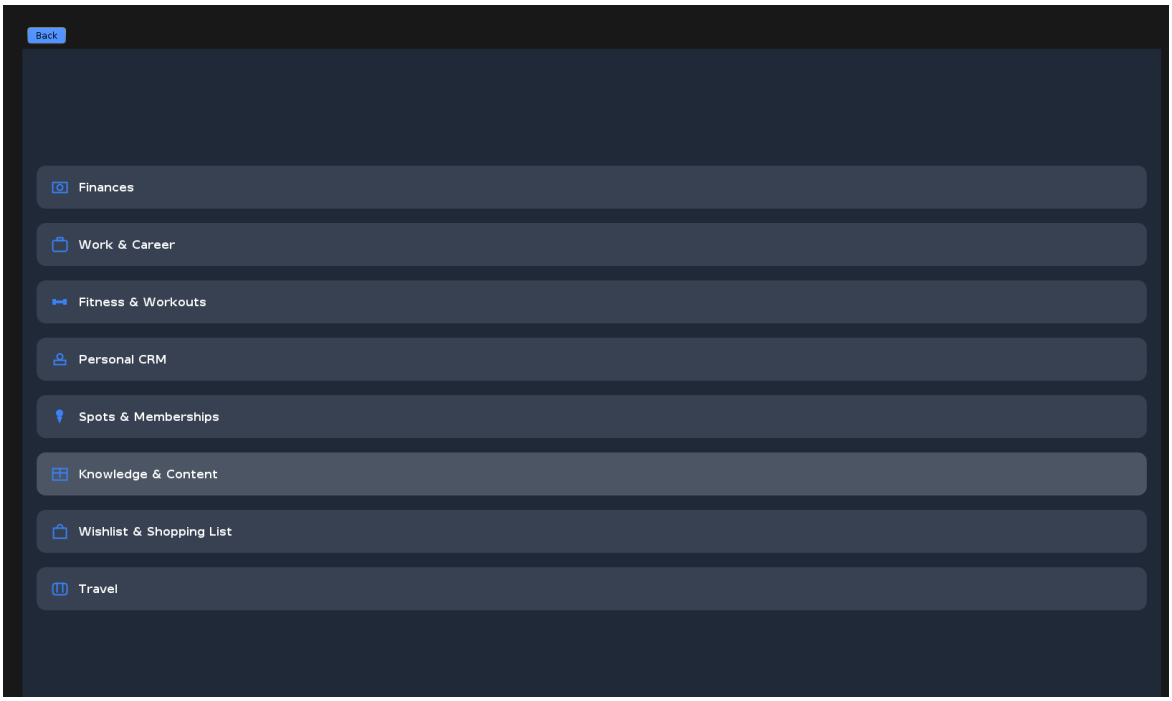


Figure 5.20: Inbuilt Template 3

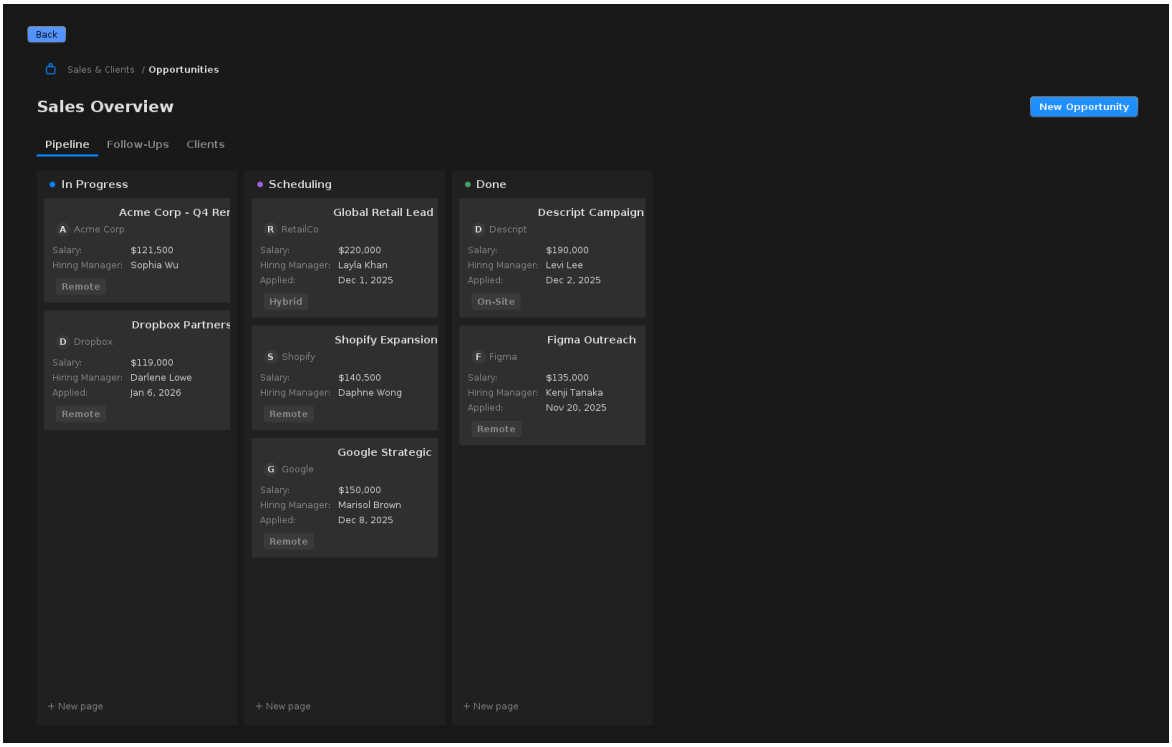


Figure 5.21: Inbuilt Template 4

CHAPTER 6

CONCLUSION

The Sol Flow system represents a modern and efficient approach to managing solo business workflows in an integrated digital environment. It combines productivity, organization, and automation into a single, distraction-free workspace tailored specifically for solopreneurs. By implementing key modules such as the Daily Planner, Custom Spreadsheet, Mail Organizer, Document Upload Center, Client Manager, and Workflow Marketplace, the system enables users to streamline their daily operations and maintain consistent productivity.

Through the use of Object-Oriented Programming (OOP) principles, Sol Flow achieves a modular, scalable, and maintainable design — ensuring that each component functions independently yet contributes cohesively to the overall system. The incorporation of reminders, notifications, and an intuitive drag-and-drop interface further enhances the user experience by improving focus and reducing friction in workflow management.

In conclusion, Sol Flow serves as a comprehensive and future-ready productivity solution for solopreneurs. It not only simplifies task and client management but also creates opportunities for collaboration and monetization through workflow sharing. The project successfully bridges the gap between multiple business tools, offering a unified, intelligent, and adaptive workspace that empowers independent professionals to work smarter and more efficiently.