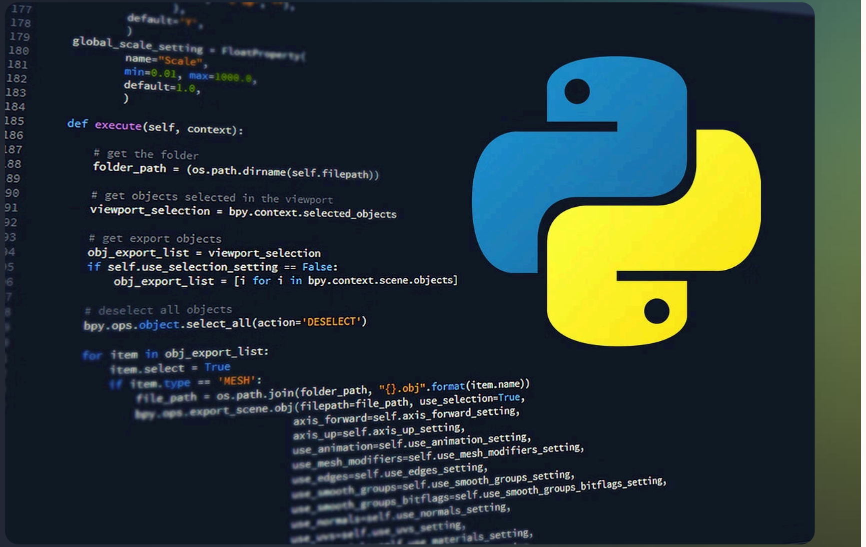


```
                ),
            default='Y',
            )
    global_scale_setting = FloatProperty(
            name="Scale",
            min=0.01, max=1000.0,
            default=1.0,
            )

    def execute(self, context):

        # get the folder
        folder_path = (os.path.dirname(self.filepath))

        # get objects selected in the viewport
        viewport_selection = bpy.context.selected_objects

        # get export objects
        obj_export_list = viewport_selection
        if self.use_selection_setting == False:
            obj_export_list = [i for i in bpy.context.scene.objects]

        # deselect all objects
        bpy.ops.object.select_all(action='DESELECT')

        for item in obj_export_list:
            item.select = True
            if item.type == 'MESH':
                file_path = os.path.join(folder_path, "{}.obj".format(item.name))
                bpy.ops.export_scene.obj(filepath=file_path, use_selection=True,
                                        axis_forward=self.axis_forward_setting,
                                        axis_up=self.axis_up_setting,
                                        use_animation=self.use_animation_setting,
                                        use_mesh_modifiers=self.use_mesh_modifiers_setting,
                                        use_edges=self.use_edges_setting,
                                        use_smooth_groups=self.use_smooth_groups_setting,
                                        use_smooth_groups_bitflags=self.use_smooth_groups_bitflags_setting,
                                        use_normals=self.use_normals_setting,
                                        use_uvs=self.use_uvs_setting,
                                        use_materials=self.use_materials_setting,
```

Introduction to To Do List with Python

Unleash your productivity with a personalized todo list built using the power of Python. Streamline your tasks, stay organized, and boost your efficiency - all with the versatility of this programming language.

Benefits of Using Python for Todo Lists

1 **Customization**

Python's flexibility allows you to tailor your todo list app to your unique needs and preferences.

2 **Automation**

Automate repetitive tasks and integrate reminders for a seamless task management experience.

3 **Cross-Platform**

Python-powered todo lists can be accessed from various devices, ensuring your tasks are always at hand.

Key Features

Task Addition

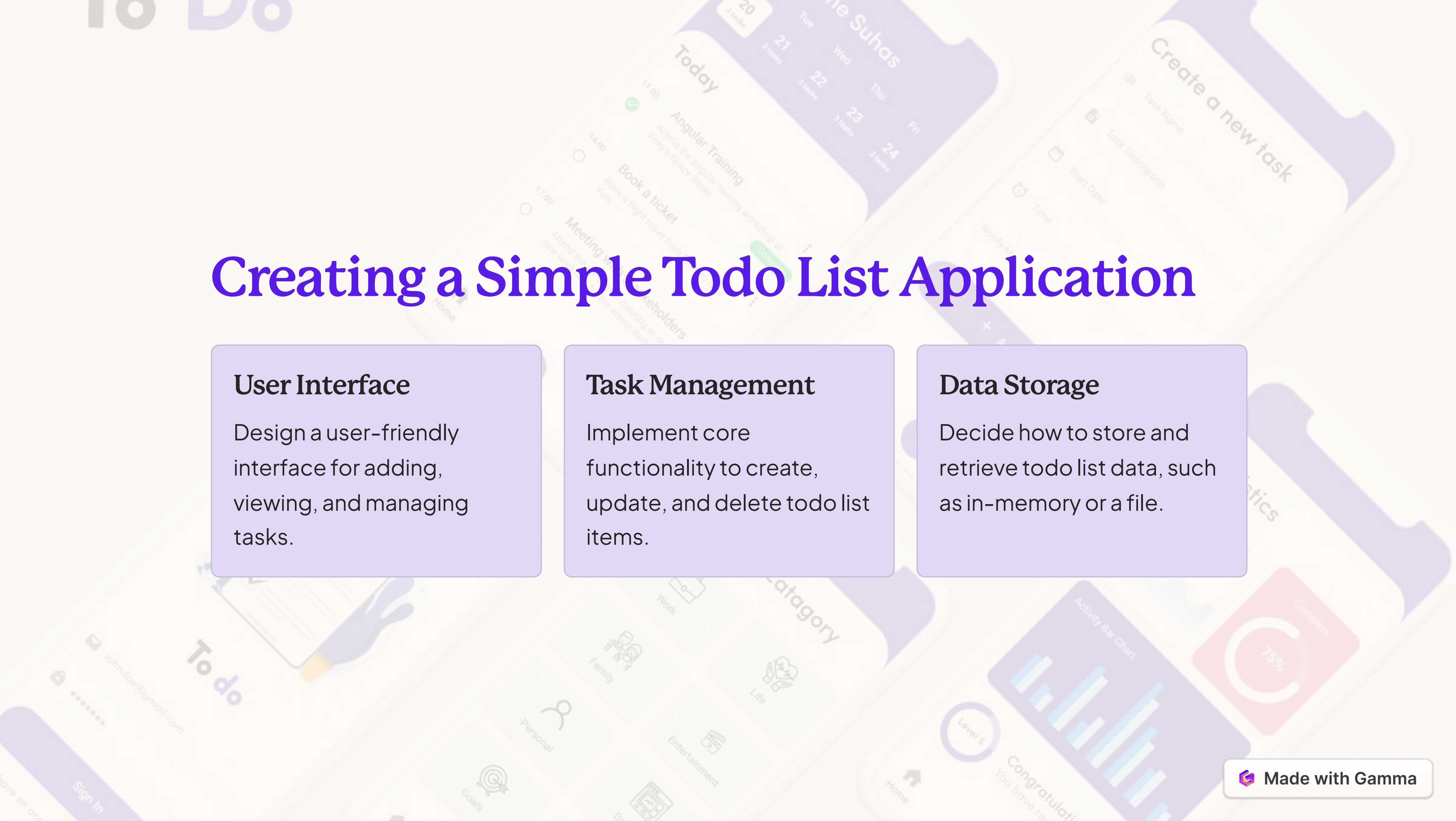
Users can easily add tasks to the system by entering a brief description of the task. The program allows for quick input, enabling users to capture tasks as they come to mind without disruption.

Task Removal

: The Automated Task Manager provides a straightforward method for removing tasks from the list. Users can specify the task they want to delete, and the program will promptly remove it from the task list

Task Listing

The program offers a comprehensive view of all tasks currently in the system. Tasks are displayed in a structured format, making it easy for users to review their task list and identify priorities.



Creating a Simple Todo List Application

User Interface

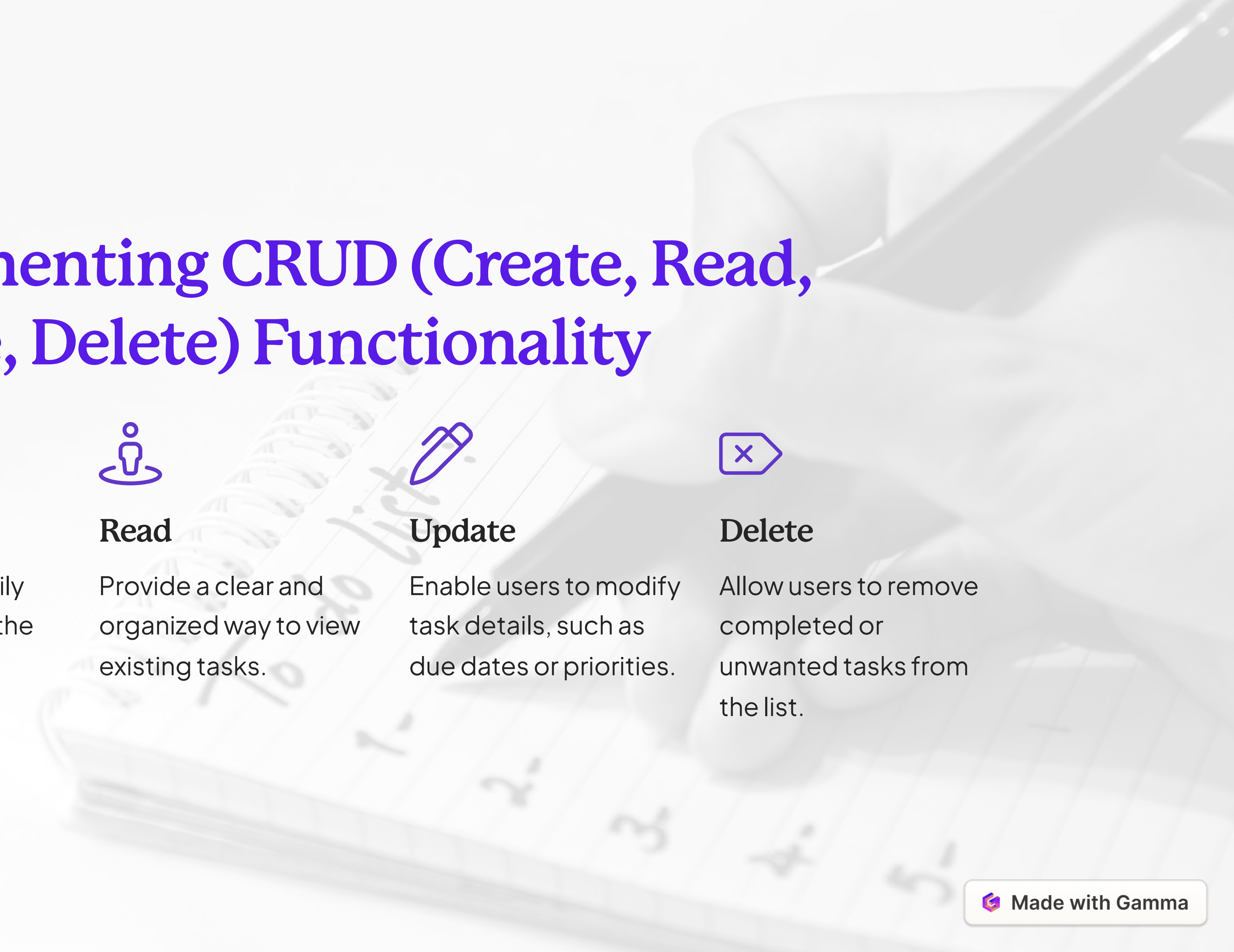
Design a user-friendly interface for adding, viewing, and managing tasks.

Task Management

Implement core functionality to create, update, and delete todo list items.

Data Storage

Decide how to store and retrieve todo list data, such as in-memory or a file.



Implementing CRUD (Create, Read, Update, Delete) Functionality

Create

Allow users to easily add new tasks to the todo list.

Read

Provide a clear and organized way to view existing tasks.

Update

Enable users to modify task details, such as due dates or priorities.

Delete

Allow users to remove completed or unwanted tasks from the list.

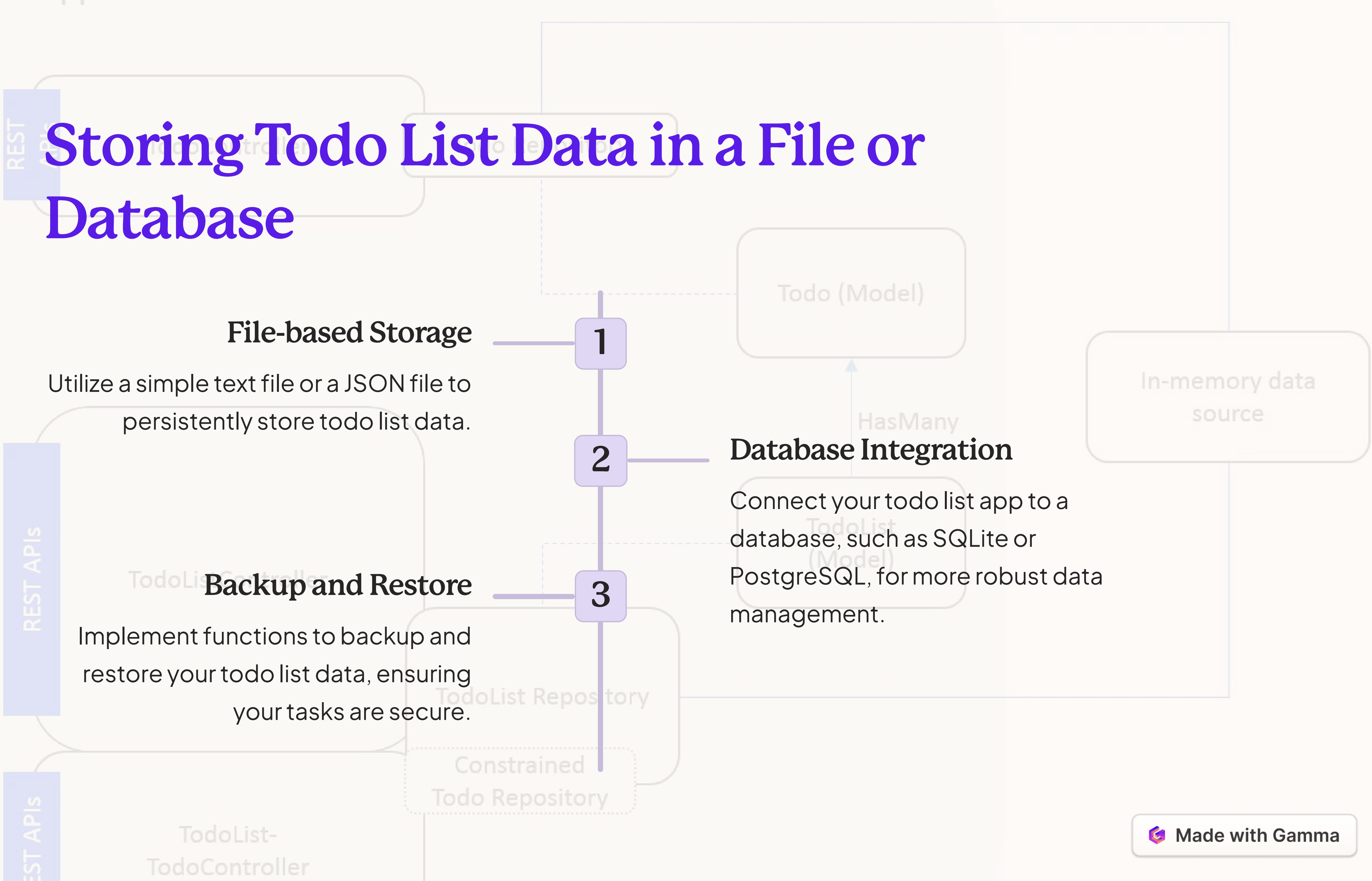
Examples:

Google Task

Microsoft To Do

Todoist

Storing Todo List Data in a File or Database



File-based Storage

1

Utilize a simple text file or a JSON file to persistently store todo list data.

Database Integration

2

Connect your todo list app to a database, such as SQLite or PostgreSQL, for more robust data management.

Backup and Restore

3

Implement functions to backup and restore your todo list data, ensuring your tasks are secure.

Made with Gamma

Enhancing the Todo List with Reminders and Notifications

1

Due Date Reminders

Set up automated reminders to notify users of upcoming task deadlines.

2

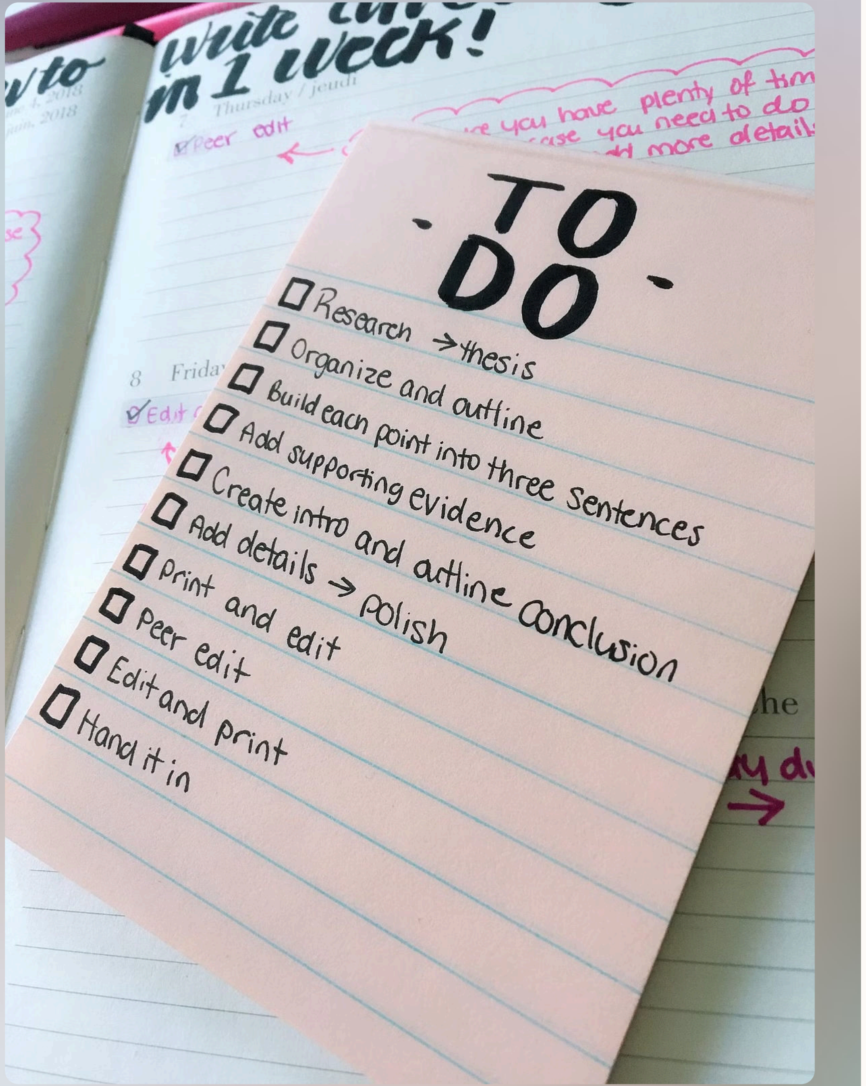
Task Notifications

Send push notifications or email alerts to keep users informed about their tasks.

3

Customizable Settings

Allow users to personalize the frequency and delivery method of their reminders and notifications.



Conclusion and Next Steps

Key Takeaways	Unleash the power of Python to build a customized todo list app that boosts your productivity.
Next Steps	Explore advanced features like project management, collaboration, and integration with other tools.