Capture and Organize

Project members:

FNU Maria – 2873052

Nazia Hassan– 2911095

DEVELOPER GUIDE

1. Project Overview

1.1 Architecture

Capture and Organize follows a **Model-View-Controller (MVC)** architecture:

- **Models**: Manage data and business logic (TaskModel, EventModel, UserSettingsModel)
- **Views**: Handle UI rendering and user interactions (TaskView, EventView, CalendarView, SettingsView)
- **Controllers**: Coordinate between Models and Views (TaskController, EventController, UserSettingsController)

1.2 Technology Stack

- **Frontend**: HTML5, CSS3, Vanilla JavaScript (ES6+)
- **OCR Engine**: Tesseract.js v4 (client-side)
- **Storage**: Browser Local Storage API
- **External APIs**: MediaDevices API (camera), Notification API, Google Calendar URL API

2. Project Structure

text

```
capture-and-organize/
├── index.html          # Main HTML file
├── css/
│   └── style.css
├── js/
│   ├── app.js          # Application entry point
│   ├── models/
│   │   ├── TaskModel.js
│   │   ├── EventModel.js
│   │   └── UserSettingsModel.js
│   ├── views/          # View components
│   │   ├── TaskView.js
│   │   ├── EventView.js
│   │   ├── CalendarView.js
│   │   └── SettingsView.js
│   └── controllers/
│       ├── TaskController.js
│       ├── EventController.js
│       └── UserSettingsController.js
└── README.md
```

3. Getting Started

3.1 Prerequisites

- Modern web browser (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
- Code editor (VS Code recommended)
- Local server for development

3.2 Setup Instructions

bash

```
# Clone or create project directory
mkdir capture-and-organize
cd capture-and-organize

# Create folder structure
mkdir -p css js/models js/views js/controllers

# Create HTML file
touch index.html
touch css/style.css
touch js/app.js

# Create model files
touch js/models/TaskModel.js
touch js/models/EventModel.js
touch js/models/UserSettingsModel.js

# Create view files
touch js/views/TaskView.js
```

touch js/views/EventView.js

touch js/views/CalendarView.js

touch js/views/SettingsView.js

Create controller files

touch js/controllers/TaskController.js

touch js/controllers/EventController.js

touch js/controllers/UserSettingsController.js

3.3 Running the Application

Option 1: Python HTTP Server

bash

python -m http.server 8000

Open http://localhost:8000

Option 2: VS Code Live Server

- Install "Live Server" extension
- Right-click index.html → "Open with Live Server"

Option 3: Node.js http-server

bash

npx http-server

Open http://localhost:8080

4. Data Models

4.1 TaskModel

Purpose: Manages task data persistence
Methods:

- addTask(title, priority): Creates new task

- updateTask(id, updates): Updates existing task
- deleteTask(id): Removes task
- getTasks(): Returns all tasks
- clearAllTasks(): Clears all tasks

Data Structure:

javascript

```
{
  id: "string",
  title: "string",
  priority: "low|medium|high",
  status: "pending|done",
  dateCreated: "ISO string",
  dueDate: "ISO string|null"
}
```

4.2 EventModel

Purpose: Manages event data persistence
Methods:

- addEvent(title, date, description, image, source): Creates new event
- deleteEvent(id): Removes event
- getEvents(): Returns all events (sorted by date)
- clearAllEvents(): Clears all events

Data Structure:

javascript

```
{
  id: "string",
  title: "string",
```

```
 date: "ISO string",
 description: "string",
 image: "base64|null",
 source: "OCR|manual",
 createdAt: "ISO string"
}
```

4.3 UserSettingsModel

Purpose: Manages user preferences
Methods:

- saveSettings(settings): Saves theme, font size, notifications
- getSettings(): Returns current settings
- applySettings(): Applies settings to DOM
- clearAllData(): Clears all application data

Data Structure:

javascript

```
{
 theme: "light|dark",
 fontSize: "12|14|16|18",
 notifications: true|false
}
```

5. Views

5.1 TaskView

Elements:

- #task-list: Container for task items
- #task-title: Input for task title

- #task-priority: Select for priority level
- #add-task-btn: Button to add task

Methods:

- bindAddTask(handler): Binds add task functionality
- bindDeleteTask(handler): Binds delete functionality
- bindUpdateTaskStatus(handler): Binds status update
- displayTasks(tasks): Renders task list

5.2 EventView

Elements:

- #camera-preview: Video element for camera feed
- #capture-btn: Button to capture photo
- #extract-text-btn: Button to process OCR
- #extracted-text: Display for OCR results
- #event-title, #event-datetime, #event-description: Form inputs
- #save-local-btn, #export-google-btn: Save buttons

Methods:

- bindCaptureImage(handler): Binds image capture
- bindExtractText(handler): Binds OCR processing
- showCameraPreview(stream): Displays camera feed
- captureImageFromCamera(): Captures image from video

5.3 CalendarView

Elements:

- #calendar: Container for calendar events

Methods:

- displayCalendar(events): Renders events in calendar

5.4 SettingsView

Elements:

- #theme-select: Theme selector
- #font-size: Font size selector
- #notifications-toggle: Notifications toggle
- #save-settings-btn: Save settings button
- #clear-data-btn: Clear data button

Methods:

- bindSaveSettings(handler): Binds settings save
- bindClearData(handler): Binds data clearing
- displaySettings(settings): Populates settings form

6. Controllers

6.1 TaskController

Responsibilities:

- Handle task creation, updates, deletion
- Validate task inputs
- Coordinate between TaskModel and TaskView

Methods:

- handleAddTask(title, priority): Processes new task
- handleDeleteTask(id): Processes task deletion
- handleUpdateTaskStatus(id, status): Processes status change

6.2 EventController

Responsibilities:

- Manage camera access and image capture
- Handle OCR text extraction
- Process event creation and export

- Validate event data

Key OCR Methods:

- extractDateTime(text): Parses dates from OCR text
- autoFillEventForm(text): Auto-populates form fields
- handleExportGoogleEvent(eventData): Generates Google Calendar URL

6.3 UserSettingsController

Responsibilities:

- Handle user preference changes
- Manage data clearing operations
- Apply settings to application

7. OCR Integration

7.1 Tesseract.js Configuration

```
javascript
const result = await Tesseract.recognize(
  imageData,      // Base64 or Image element
  'eng',          // Language
  {
    logger: m => console.log(m) // Progress logger
  }
);
```

7.2 Date Parsing Patterns

The system supports multiple date formats:

- MM/DD/YYYY, MM-DD-YYYY
- DD/MM/YYYY, DD-MM-YYYY
- YYYY/MM/DD, YYYY-MM-DD

- Month names (December 15, 2025)
- Relative dates (today, tomorrow)

8. Local Storage Management

8.1 Storage Keys

javascript

```
localStorage.setItem('tasks', JSON.stringify(tasks));
localStorage.setItem('events', JSON.stringify(events));
localStorage.setItem('userSettings', JSON.stringify(settings));
```

8.2 Storage Limitations

- Maximum 5-10MB per origin
- Data persists across browser sessions
- Cleared when user clears browser data

9. Testing Guidelines

9.1 Unit Testing

Test individual components in isolation:

- Model methods with mocked localStorage
- View rendering with sample data
- Controller logic with stubbed dependencies

9.2 Integration Testing

Test component interactions:

- MVC data flow
- Camera to OCR pipeline
- Local storage integration

9.3 Browser Testing

Test across browsers:

- Chrome, Firefox, Safari, Edge
- Mobile and desktop views
- Different screen sizes

10. Debugging Tips

10.1 Common Issues

1. **Camera not working**: Check browser permissions
2. **OCR failing**: Verify image quality and network connection
3. **Data not persisting**: Check localStorage quotas
4. **UI not updating**: Verify event listeners are bound

10.2 Debug Tools

- Browser Developer Tools (F12)
- Console logging in controllers
- localStorage inspection
- Network tab for OCR requests

11. Deployment

11.1 Hosting Options

Free Hosting:

- GitHub Pages
- Netlify : https://capture-and-organize.netlify.app/

11.2 Deployment Steps

For GitHub Pages

git init

git add .

```
git commit -m "Initial commit"
git branch -M main
git remote add origin https://github.com/username/repo.git
git push -u origin main
# Enable GitHub Pages in repository settings
```

12. Future Enhancements

12.1 Priority Features

1. **Reminder Notifications**: Browser Notification API integration
2. **Task Categories**: Organize tasks by category/tag
3. **Event Recurrence**: Support for recurring events
4. **Data Export**: Export tasks/events as CSV or JSON

12.2 Technical Improvements

1. **Service Worker**: Enable offline functionality
2. **IndexedDB**: Replace localStorage for larger data
3. **Progressive Web App**: Add PWA capabilities
4. **Accessibility**: Improve screen reader support