Agenda

- Rasmus Jonsson – LibreSign
- Shubham Goyal – QR Code
- Sumit Chauhan – Notebookbar
- Ahmed ElShreif – Improve UI testing with new DSL
- Kaishu Sahu – LibreOffice Android App improvements
- (*) Gagandeep Singh - Implement Chart Styles

* slides missing
LibreOffice Appliances

Rasmus Jonsson - wasmus@zom.bi

- Overview
- Features
  - LibreSign
  - Control panel
  - Web Impress Remote
- Status
• Goal: LibreOffice Impress presentations running automatically on a screen (digital signage)
• Plan and implementation open-ended
• Result: Three part project, LibreOffice controller, control panel, web remote
LibreSign

- Main program, available as a Python package
  - `pip install libresign`
- Starts up LibreOffice and web servers
- Loads and plays ODP files provided by users in LibreOffice
Control panel

• Single page application
  – Upload and download presentations
  – Arrange the “playlist”
  – Select presentation to play

• Accessible on local network, link on screen when running LibreSign
Web Impress Remote

- Single page application
  - Modelled after the Android Impress Remote
  - Slide previews and notes
  - Select slide to display
- Accessible from control panel
Features to be added
  - Mostly visual / user experience
  - Some optimisations

Available for download on pypi.org and github.com

Usable
QR Code Generator in LibreOffice

Shubham Goyal -
22shubh22@gmail.com

- Project Description
- Work Done
- Work Remaining
- Glimpse of Work
- Student’s LibreOffice Future Plans
Project: QR Code Generator

Description / Task:

1) Integrate a QR Code generating Library @ LibreOffice

2) Provide a platform (Dialog Box) to take user inputs required for generating QR Code

3) Handle the QR Code as a shape in LibreOffice

4) Integrate the QR Code feature with LO apps.
Project: QR Code Generator

- QR Code Feature
  - ODF import/export: QR Code can retain its property and it can also be edited.
  - QR Code generated as SVG.
To-Do: QR Code Generator

- Disable other shape features like side scaling which can accidentally deform the QR Code. (ticket: tdf#127460)
- Write OOXML changes for QR Code. Currently QR Code property is lost while save/load in case of odp.
- Regression (tdf#126631):
  - Occurred from one of my commits.
  - Can’t verify the crash stated.
**Insert → QR Code..**

---

**QR Code Dialog**

**Generated QR Code**
Door Openings

• tdf#87195 - mail merge: add QR from URL field
Future plans with LibreOffice

• Work on the previous To-Do task.
• Work on enhancing chart wizard with Pedro.
• Voluntarily solve issues.
• Participate in upcoming hack-fests.
● Customization Support
   - Customization Support allows users to customize Notebookbar by changing the visibility (show/hide) of buttons.

● Extension Support
   - Extension Support allows users to add extensions in Notebookbar. There is an extension tab in all the Notebookbar interfaces where the added extension will be available.
What’s done:

- A basic Customization that would allow the user to show/hide items in the Notebookbar. You can find the dialog at Tools->Customize -> Notebookbar(tab)
How are things done:

- We stored and reload the modified UI files in/from the user directory. We also stored the modified data in registrymodifications.xcu so that configurations can be retrieved if there is a version update.

- Libxml2 is used to parse the UI files.

- The class \texttt{SvxNotebookbarConfigPage} is inherited from \texttt{SvxConfigPage}. \texttt{SvxConfigPage} is a base class for all the customization dialog in LibreOffice.

Future Work:

- There are many features still locked and need to finish.

- The use registrymodifications.xcu to retrieve the configuration and use them during an update is not complete.

- It would be better to open the customization dialog based on the tab opened in the Notebookbar interface.
What’s done:

- The task to add extension under the extension tab is finished.
- Extension developers can now create extension for Notebookbar.
How are things done:

- Schema is added for adding extension under extension tab and anywhere in Notebookbar.
  
- Code in `addonoptions.cxx` is extended for reading above schema from `addons.xcu` file.
  
- Extensions are rendered in UI during runtime. The class `NotebookBarAddonsMerger` is called inside the method of class `VclBuilder`.
  
- Reduction in runtime and redundancy is done by adding code in `vcl` directory.

Details: [https://wiki.documentfoundation.org/Development/NotebookBar/Extensions](https://wiki.documentfoundation.org/Development/NotebookBar/Extensions)

Future Work:

- The task to add extension anywhere in the Notebookbar
  
- Create a new custom widget namely `sfxlo-extension` for adding extension.
Improve UI testing with new DSL

Ahmed ElShreif - aelshreif7@gmail.com

- Introduction
  - What is DSL and Why to use it?
- Implementation
  - Tools used in the new DSL.
- Logging System
  - What is logging system?
- Usage
  - How to Use this project
- Feedback
  - Some resources and documentations
New DSL

- **DSL?**
  it’s just a new programming language that can be used with new defined syntax.

- **Why?**
  it really make it easy for understanding the code and use it without programming background.

- **Implementation?**
  we used TextX{python library} to implement the new syntax with the new Compiler.
• Logging?
  Most of the main events are now supported and logged and be able to compile.

• Why this is important?
  We can now just open libreOffice and log some actions then we can automatically generate a UI test that replay all these actions with python UI framework.

• Documentation?
  please try to see this blog: GSOC19 Ahmed ElShreif on blogger
How to use?

1) Launch LibreOffice and enable logging option
2) Simulate what you want to do with the mouse
3) Close LibreOffice
4) Open the resulting file that you choose to log actions in
5) Enter the UI logger directory
6) Use the following Command

python dsl_core.py <path_to_log_file> <path_to_a_new_python_file>
Kaishu Sahu: LibreOffice Android App improvements
• **New features**
  - **Print support:** One can print the document with the printer connected to the same network.
  - **SlideShow support:** The user can view the document in the full-screen mode.
  - **Insert Image in the document:** This allows to insert the image in the document from the phone’s gallery.
  - **Share document:** The user can use the existing messaging applications to share the document from the LibreOffice.
  - **Save_As the document:** By this, one can make the fresh copy of the document anywhere in the storage media.
  - **Rationale dialog for permissions:** To give the better understanding of a particular permission.
  - **Launcher shortcuts support for api > 24:** The user can open the recent documents from the launcher (only if the launcher supports the shortcut feature).
  - **Support for other document formats in save_as and share feature:** The user can choose the format of the document in which they want to share/save it.
  - **Dimming the document when inactive.**
• Minor Changes:
  - addition of no_recent_items/no_items msg.
  - conversion of the remaining explicit intents to implicit intents.
  - preservation of the state of the MainActivity upon restart
  - replacement of PreferenceFragment with PreferenceFragmentCompat (because the former one is deprecated)
  - addition of cut, copy, paste support to document viewer.
  - addition of the support for screen dimming from JS.

• Bugs fixes:
  - order of elements in 'AndroidManifest.xml' as per guidelines and icon of the android app wasn't reflecting in the launcher
  - setting all default values to setting preference
  - ArrayIndexOutOfBoundsException on selecting 'Drawings' in DefaultFileFilter setting
  - update handling of the storage permission.
  - keyboard popping out on clicking edit button.
  - contextToolbar's cut, paste without editing permission bug.
Many thanks to Google, and all the mentors!