DIALED IN: FINAL REPORT

ABSTRACT

The Dialed In application is a task tracking app for groups. It's built on the Windows Phone 7 platform and enables multiple users, organized into a group, to view the same set of tasks and assign these tasks to each other. In addition, it allows for tracking some rich task data that is unique to mobile devices: location (GPS), photos & video (phone camera). It is targeted at families who want to track tasks for all family members and other project-based groups who want to be able to track tasks collectively. Users have the ability to belong to multiple groups along with their own personal task group. This enables them to keep organized in all areas of life.

We evaluated our solution by interviewing eleven people, ten of them in work/family groups. Feedback received from these sessions was positive overall as to the concept of the app and the user experience we implemented. We learned of some areas for improvement and incorporated one of those into the final app.

INTRODUCTION

The Dialed In application helps families stay organized and on top of the things they need to do as individuals and as a family by providing them with a unique mobile app to track their tasks. Tasks lists are typically very simple with few words; little fancy formatting; and a shallow hierarchy. However, they're often used in complex ways by sharing items, capturing tasks on the go and expecting them to be relevant to the current time and place. Dialed In will allow users to take advantage of what mobile technology has to offer in the world of tasks.

Dialed In will succeed as a solution to this problem because it is targeted at the most simple scenarios while still providing richness of functionality. We don't add unnecessary features that will only serve to distract from the user's ultimate goal of staying in control of his life. Our most significant competitor is "Remember the Milk", a full-featured solution to task management. However, "Remember the Milk" has a mobile offering only as an appendage to the full web offering, which we feel detracts from the benefits of using a mobile device. With Dialed In, we make the location of you and your tasks a central focus, making it easy to plan your day, something that is unique to a mobile device.

We chose to use the Windows Phone 7 platform for this app because of the knowledge we already have about the platform and because of the connections that we have to platform experts through our jobs. In addition, our day to day work would be better served with knowledge of the WP7 platform, so we felt it was most useful for our professional growth. Finally, Windows Phone 7 has a unique UI concept which allows us to organize the user experience around tasks very efficiently. We were excited to try building an app on top of this 'metro' UI to see how it fared with this type of information.

RELATED WORK

Ability to keep track of one's current task list through software has been a much demanded feature since the advent of personal computing. As computing power and connectivity has increased, users have, in increasing numbers, abandoned their physical day planners and agenda books in order to take advantage of the new technological capabilities. In order to be competitive, software implementations have needed to expand their feature sets beyond what their pencil-&-paper counterparts were capable of in novel ways.

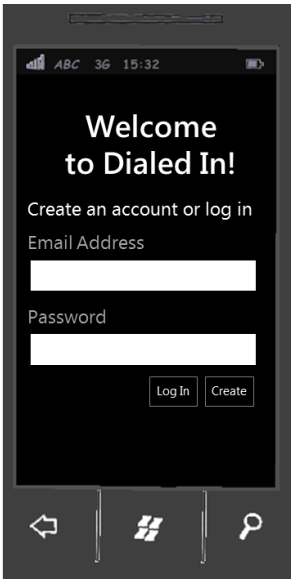
Desktop implementations, such as in Microsoft Outlook[1], have greatly expanded their feature sets as they have iterated through releases the past few decades. These programs allow the user to set recurrence, mark status (with "Not Started", "In Progress", and "Waiting on Someone Else" as some of the options available), show percentage complete, and add billing information. Tasks can also be associated with people from one's list of contacts or with an e-mail message from one's Inbox. Task sharing with other users is permitted, but only if all users have accounts on the same Microsoft Exchange server[2]. Much of the differentiation is thus oriented towards business environments – a user in a consumer environment would be unlikely to make use of them.

Recently, the growth of the web as a platform has led to more consumer-friendly implementations, such as Google Tasks and Hotmail Calendar. These provide vastly simplified interfaces and options when compared to their desktop counterparts, as well as accessibility from any computer terminal[3]; Google Tasks additionally allows the creation of sub-tasks in a list[4]. But the group functionality provided by desktop implementations is not present in these.

Many of the existing desktop and web task managers have been ported to mobile platforms, but the most unique and fully featured application is "Remember the Milk"[5]. This application, available for iOS, Android, and Blackberry as well as the desktop, permits users not only to manage their tasks as in other apps but also to coordinate with other services, such as by receiving reminders via e-mail/SMS/IM. Sharing of task lists is permitted via publishing them to others.

USAGE MODEL

INSTALL AND SETUP



At the top level, users have the application installed on a phone. Each user has an account that is used to sign into the app. The app remembers this account on a given phone. This account can be used across multiple devices. Tasks are tracked as part of groups, which allow multiple members. Each account can be a member of multiple groups. One default group is created for each account for the account owner to store personal tasks.

Users can come across the app by searching the Windows Phone Marketplace, or they can receive an e-mail invitation from another Dialed In user, pointing them to the Marketplace. Installation of the app is straightforward on a WP7 device through the Marketplace infrastructure. The app is first downloaded and then installed with the user only having to confirm once.

The first time the app is launched, the user will be asked to either sign in or create an account. If the user chooses to create an account, a default group will be created for that user to store personal tasks in; however, that group can be shared with others just like all custom-created groups.

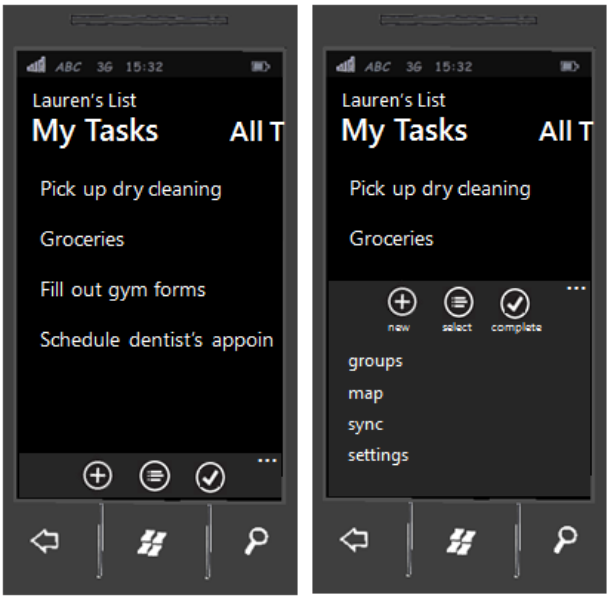
If the user creates an account with an e-mail address that has received invitations from other groups, those invitations will be associated with the account and the user will be able to accept those invitations. More details on this follow below.

GENERAL EXPERIENCE

The main app experience consists of a panel with three views: My Tasks, All Tasks, and Notifications. This panel is filtered based on which group is selected in the group selector. By default, it will show the user's personal group.

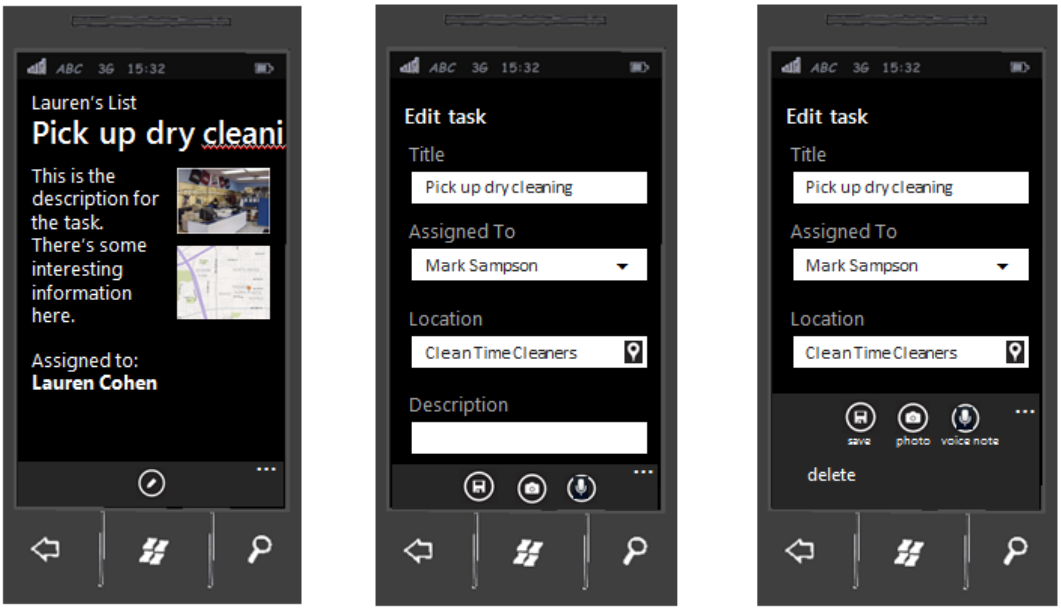
My Tasks shows all tasks in that group which are assigned to the user. All Tasks includes all tasks assigned to others. The Notifications view shows information about changes to tasks. For example, if a task has recently been assigned to the current user, a notification would appear so that the user knows there's a new item in his list.

There are a number of things that can be done from this view: viewing (and editing) existing task details, creating new tasks, selecting and marking tasks as complete, changing the group filter, plotting tasks on a map, forcing a sync, and accessing app settings. Each of these is described in more detail below.



VIEWING/EDITING A TASK

Viewing a task is done by touching the task itself. The user can then edit it by touching the 'pen' button on the bottom of the task's view. Editing is done in the same context that the user uses for creating a new task. Viewing a task shows all the details of a task including some visuals of the image and location associated with that task.



When any changes are made, the user touches Save to commit them. A task can also be deleted from the Edit page.

ADDING A TASK

Adding a task is done by touching the '+' button on the bottom of the My Tasks view. This brings up the task form which allows the user to add a title and other details to the task. Task metadata includes: "Title", "Assigned To", "Location", "Description", and potentially a photo. The user only needs to enter the Title of the task; reasonable defaults will be assigned to the other fields or they will be left blank. For example, "Assigned" To will by default point to the current person; however, location will be left blank.

For setting the task location, the user can type a manual location into the location field, or, by clicking the map button next to the field, the user can select a particular location on a map.

MARKING TASKS AS COMPLETE

One or more tasks can be selected at the same time using the select button to show checkboxes and then checking each desired task. Or, the check box can be shown by touching to the left of a task. This is a standard WP7 metaphor.

When a task is selected, the user can choose to mark it as complete. The Complete button is not available unless a task is selected.

CHANGING THE GROUP FILTER

The groups which the current user is a member of are all available for viewing by accessing the groups filter. Here, the user can select a single group, or can choose to view tasks from all groups at once. When group is selected, the following group selector appears:

VIEWING TASKS ON A MAP

The user can choose to view tasks on a map by clicking the 'map' option. The map will default to only showing the tasks in the context of where the map was accessed from, but the user can choose to change the filter to show a different set of tasks. For example, if the map is accessed from My Tasks, the user will initially only see his own tasks. If the user had a number of tasks selected, only those tasks would show on the map.

FORCING A SYNC

Users can choose to force a sync with the back end at any time, though there is also an automatic sync which happens periodically based on the setting the user has chosen. The default is every 15 minutes.

ACCESSING APP SETTINGS

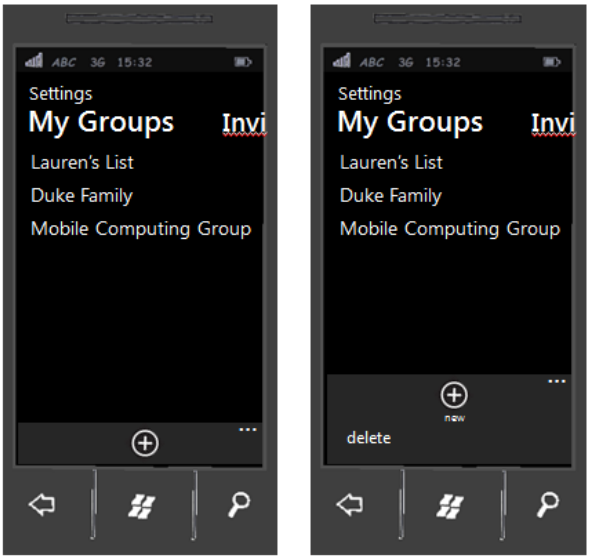
App settings are described in more detail below. They are accessed from the main tasks list.

APP SETTINGS

The application's settings consist of another panel of three pages: My Groups, Invitations and Other. My Groups shows the list of groups the user is a member of. The Invitations page shows the list of pending invitations to other groups. The Other page shows miscellaneous app options.

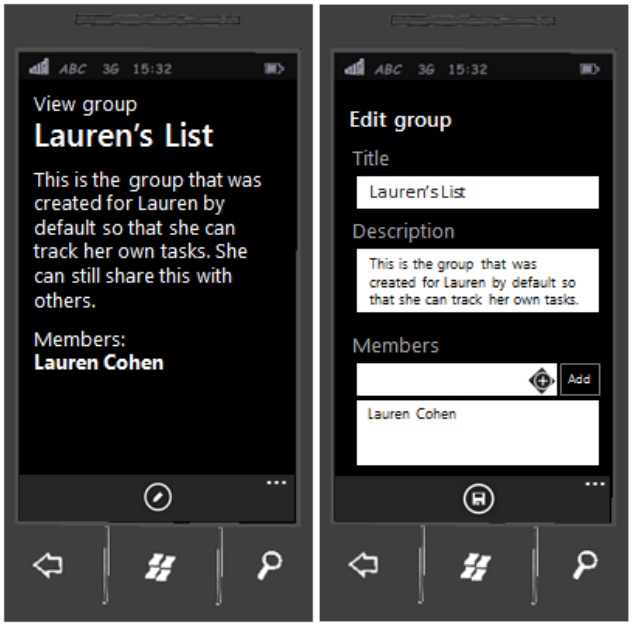
MY GROUPS

In the My Groups page, the user can access details about each group in order to add other members, or he can create a new group. Membership in a group can also be removed from here by touching delete when a group is selected. A confirmation would appear so the user doesn't inadvertently remove groups.



To view the details of a group, the user would tap on a group's name. This would show the detailed information about the group along with providing the option to edit that group. Editing the group uses the same view as adding a new group.

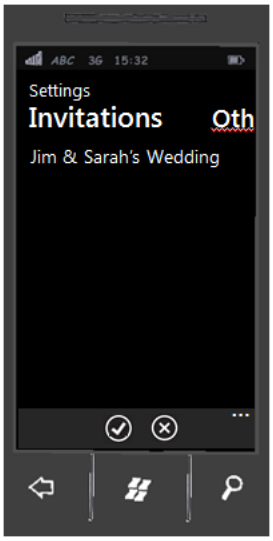
The user can change the title or description for a group along with the list of members. To add new members to a group, the user can type in an email address or select a user from the phone's directory. When a person from the phone's directory or an email address is present in the text box, tapping Add will place that person into the members list.



Tapping save commits any changes. For simplicity, any member of a group can make modifications to the group.

INVITATIONS

When a user is added to a group, an invitation to that group is associated with that user's account. The user can see those invitations and choose to accept or decline them on his or her phone. Once an invitation is accepted, that group will end up in the user's My Groups list.



OTHER

Finally, the sync frequency can be changed on the Other page by choosing an option from a drop-down. By default it is set to sync every 15 minutes, but the user can choose to never sync or can select a number of other durations.

ARCHITECTURE DESIGN

CLIENT ARCHITECTURE

This section describes the client application for the phone as implemented. Features not implemented (mainly due to lack of time) are not described in this section.

DESIGN OVERVIEW

The Dialed In client application is a Silverlight application that follows the Model-View-ViewModel (MVVM) design pattern**.** The MVVM pattern is very similar to the popular Model-View-Controller pattern. The MVVM pattern is helps the developer abstract the data from the user interface and results in a clean design with lots of reusable components.

The model represents the application data. In general, the model only contains definitions of data and methods for accessing data. The model is defined as a set of c# classes grouped into a namespace.

The view represents the application screens. Each view consists of XAML file which defines the elements in the application screen and a code behind file which contains the logic for the application screen including displaying and saving data, event handlers for UI controls such as buttons and text boxes and system API calls for navigating to other applications screens.

The view model links the application screens to the application's data. The view model instantiates and manages the data for the application. The view model is defined in a c# class that is it's own namespace. The application contains one instance of the view model class that is defined as a static variable of the application base page.

MODEL

The model is defined in the DialedIn.Model namespace. The following classes are defined in this namespace.

- Class AppInstance;
  - Contains application instance specific data
- Class Task;
  - Contains data for a single task
- Class Group;
  - Contains a group data including a collection of Tasks
- Class Notification
  - Contains notification data

VIEW MODEL

The view model is defined in the DialedIn.ViewModel namespace.

The view model class defines a collection of group objects which each contain a collection of tasks. The view model also defines a collection of notifications.

The view model name space contains one class that defines private helper methods and the following public methods:

- GetGroups()
  - Initializes list of groups from phone or storage or creates default groups if persisted data is not available
- SetSelectedGroup()
  - Sets the currently selected group for the application
- GetGroupWithTitle()
  - Returns the group object if one is found that matches the title argument
- SetSelectedTeask()
  - Sets the currently selected task for the application
- AddGroup()
- AddTask()
- AddNotification()

VIEW

The application consists of the following seven screens.

LOGIN SCREEN

The login page is where the user enters their credentials. In the current implementation, only the user name field is stored and used for assigning the default tasks and groups. In a production implementation of the application, username and password would actually be used for authentication.

This screen contains a button control for submitting the username and password and entering the application. The code behind for this screen contains a click event handler, which controls saving the username and navigating to the initial application screen.

TASK/NOTIFICATION OVERVIEW SCREEN

The task overview screen follows the pivot page design template, which provides a convenient way to view different configurations of a set of data. This screen display tasks assigned to the current user for the current group in one pivot, all tasks in another pivot, and notifications in the final pivot.

Each pivot contains a listbox. The listbox is bound to the collection of tasks in the view model in the XAML file. The ability to bind an object allows for only having to create 1 template for all of the listbox items. LINQ queries are used to select relevant items from the collection to the list box. LINQ is another very useful feature of c# that makes search a collection of elements very easy and clean.

The screen has a button for adding new tasks and a context menu, which provides access to the groups page. The user can also tap directly on the task list box item to see a detailed view of that item.

TASK DETAILS SCREEN

The task details screen provides a view of all of the data in the task. The assigned to, due date, location, task title and task description are displayed in this page.

This screen has a button with a check mark. This button sets the task to completed by setting the tasks is active field to false and navigates back to the task overview screen. Data is refreshed to only include active tasks for the task overview screen.

This screen also has a button with a pencil. This button navigates to the edit screen and provides the title of the task as a parameter in the URI for the edit screen.

TASK EDIT SCREEN

The task edit screen will allows the user to edit data for an existing task or input data for a new task. This includes title, assigned to, location description and due date.

The assigned to text box has a plus button next to it. Clicking this button calls into the phone's chooser feature for selecting contacts entered in the phone address book. The chooser displays a view of the phone's address book. When the user taps on a contact, the application navigates back to the task edit screen. This provides a very quick input mechanism if the address is found in the phone book.

The button with the camera icon also makes use of one of the phones built in chooser elements. This chooser brings up the camera experience for the phone and brings the image back to the app for saving. When the user clicks this button and takes a photo, it is assigned to the task.

The button with the disk icon saves the task when clicked and returns the user to the task overview screen.

GROUP OVERVIEW SCREEN

The group overview screen display a list of the groups the user is a member of. If the user taps a group name, they are brought to the group details screen.

This screen contains a button with a plus icon that when clicked brings the user to the group edit screen where they can add the information for a new group.

GROUP DETAILS SCREEN

The group details screen displays the group description, owner and members. The button with the checkmark sets the group as the selected group for the application and brings the user to the tasks overview screen.

The button with the pencil navigates to the group edit page for the specific group.

GROUP EDIT SCREEN

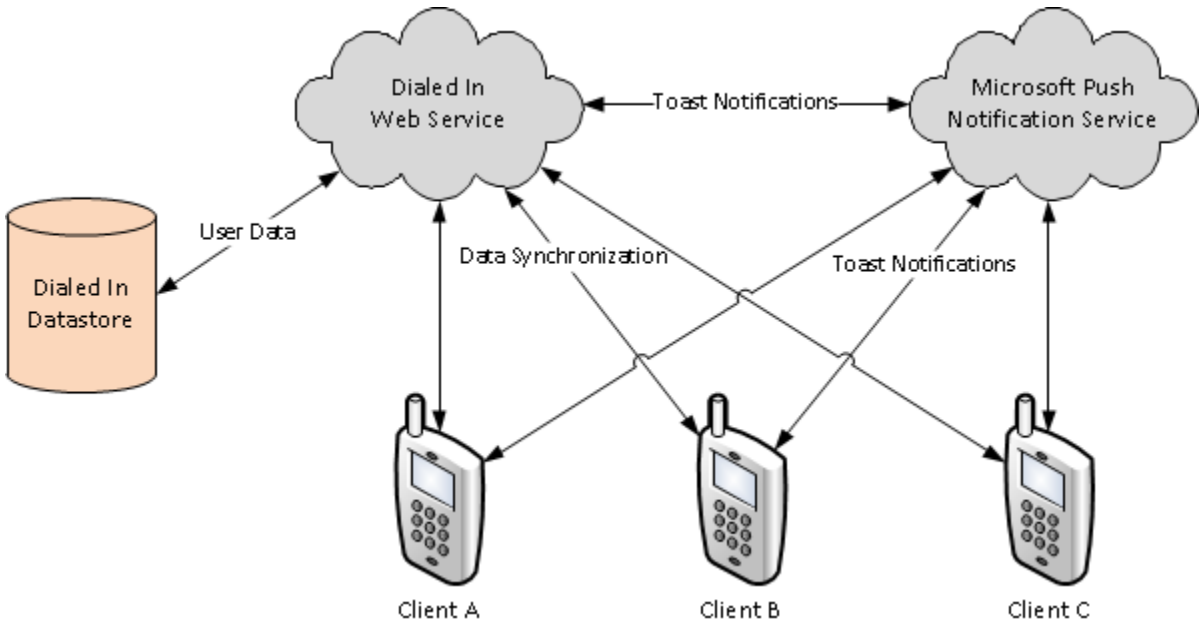
The group edit screen allows the user to change the group name and assign members to the group. This page contains the same email address chooser as the tasks edit screen making input for this field much more fluid and faster.

IDEAL BACK-END ARCHITECTURE

If we had been given unlimited time for completion of this project, we would ideally have created a full implementation of a web service to support the individual phone clients. This web service would be responsible for storing information as well as providing a connection for clients to send and receive data from.

The data store component of the web service would house all of the information; in addition to individual tasks, this would include user profiles, memberships, and list information. It would be SQL-based in order to be easily queried on several pivots.

The connectivity aspect would be constructed similarly to that of a standard e-mail service. Clients would authenticate and sync their information with our service at regular intervals using a protocol such as IMAP or POP3. One-off messages, such as task additions or edits, would be sent in response to the associated user actions to the service from the client via a protocol such as SMTP – we are assuming that users will be making a limited number of transactions (task additions/edits, group characteristic edits, etc.) in the vast majority of use sessions. Updates made by clients would be taken in chronological order and trigger a "toast", or pop-up, notification to be sent to other users following that task or group to indicate availability of an update. These "toasts" would be pushed to the client via the Microsoft Push Notification Service[11] in order to allow them to appear in UI contexts on the phone that are outside of the application itself (such as on the application's tile when pinned to the Start screen).



SIMPLIFIED BACK-END PROTOTYPE DESIGN

In order to simplify the back-end requirements for our prototype, we decided against implementing the full data store, as well as the web service components that would interface with the data store and directly with the clients. We instead focused on creating a simplified service utilizing interface with the Microsoft Push Notification Service; our belief was that enabling these basic update functions and interactions could best demonstrate the features crucial to our user experience.

NOTIFICATIONS AND LIVE TILE UPDATES

The Microsoft Push Notification Service allows for three different message types:

- Raw HTTP messages, containing basic information for the targeted application
- "Toast" notifications, which generate a pop-up banner at the top of the phone screen, regardless of the user's current activity
- "Live tile" notifications, which alter the display of the application tile displayed on the Start screen

To simulate our web service informing a client when updated information is available, we used a combination of all three message types. Our implementation specifically focused on updating the task data, but could easily be extended to handle other types of information that the server would be expected to transmit (group invites, etc.).

In order to update task data on the client, the service is first used to send a "toast" notification, which consists of a simple text string indicating the contents of the coming update. Example toasts could consist of:

- 3 tasks updated in group "Low-Resource Mobile Computing"
- New task "Pick up the dry cleaning" added to group "Family Chores"
- Task "Grocery shopping" marked completed by Jim in group "Family Chores"

This notification is processed by the client and added to the Notifications view within the Dialed In client. In addition, if the user is currently outside of the application context when the toast is received, Windows Phone 7 will play a sound and pop up a small banner (in the user's chosen UI color) at the top of the screen; pressing on this message will launch the Dialed In client. The toast persists for a set period of time (defined by the operating system) before disappearing.

In our ideal back-end implementation, launching the client from the toast message would trigger it to sync with the server and receive the new information; for prototype purposes, we instead send raw HTTP notification alongside the toast with the updated task for the client to incorporate. The task data, found in the body of the HTTP notification, is expressed in the following XML format:

```
<TaskData>
	<TaskId></TaskID>
	<GroupTitle></GroupTitle>
	<TaskTitle></TaskTitle>
	<TaskDescription></TaskDescription>
	<TaskAssignedTo></TaskAssignedTo>
	<TaskDueDate></TaskDueDate>
	<TaskLocation></TaskLocation>
	<TaskImageUrl></TaskImageUrl>
	<LastModifiedBy></LastModifiedBy>
```

```
        <LastModifiedTime></LastModifiedTime>
        <TaskFiltered></TaskFiltered>
        <TaskCompleted></TaskCompleted>
</TaskData>
```

This makes it simple for the client to parse the new data and incorporate it in to existing data structures and views. Several fields are included that are not currently implemented in the client, but would be utilized given development time for additional features.

Along with the toast and raw HTTP notifications, we also send a "live tile" notification. Receipt of this causes the Dialed In tile, if pinned to the Start screen, to change – a number, known as a "counter," in the upper right corner (as dictated by Windows Phone 7 – we unfortunately have no control over this UI aspect) will indicate how many notifications have been received since the client was last synced. This allows the user to be informed of how many updates are pending for them if they miss the corresponding toast notifications.

A limitation we found through trial-and-error of Windows Phone 7 is that a live tile notification must be sent to reset the counter to 0 to indicate that it should no longer be displayed; the client-side application does not have privileges to do this itself. Our ideal backend would send this notification automatically when synchronization is triggered between the data store and the client, but for our demonstration purposes we needed to do this manually.

To construct and send all of these notifications, we developed a Windows Communication Foundation-based web service in C#. This service currently resides on the same host as a Windows Phone 7 emulator running the Dialed In client. The client registers with the service via a port on the local host, then receive updates via a channel the service has established with the live Microsoft Push Notification Service. We construct these updates via a simple Silverlight user interface; in the absence of a data store to provide such information, the following input fields are provided along with a "Send" button:

- Drop-down selection for each XML field of task data in a raw HTTP notification
- Text string for a toast notification
- Slider to set the value of the counter in a live tile notification

The user interface also provides status on each sent notification to indicate whether it was received by the client and, if not, what errors were encountered.

Our service is capable of being deployed as written to either Windows Azure or a Windows-based server running Internet Information Services (IIS), which would break the dependency on the emulator for this feature; our only limiting factor in accomplishing this was time.

EVALUATION

To evaluate our app, we interviewed a number of users representing our target audience. We did a combination of group and couple sessions, with one individual session to start things off. The group session consisted of six participants. Following that, we had two sessions with a married couple in each. This gave us a total of eleven participants.

We intentionally stayed away from interviewing people solo (except for one initial interview) as our app is intended to be used by multiple people, so we wanted to make sure the feedback we were getting was from an

appropriate audience. The couples allowed us to see how it would be received by families while the group session allowed us to see how a project-centered group would perceive the value of the app.

We first began by taking the users through the slides we submitted as part of HW8 complemented with the actual app running on a phone for more dynamic functionality. The phone app helped users who were not familiar with the WP7 platform understand the interaction model.

After the app demo, we asked a number of questions (and reviewed the app again as necessary). The questions first focused on the overall product concept to see if users found value in what we were trying to do. Then we worked on finding out about the specific features provided and how they might or might not be useful. Finally, we looked at usability and our execution of the concepts to see if we were successful in creating an app that was straightforward and simple to use.

In all sessions, the questions generated some discussion as participants fed off of each other to clarify their positions. We feel this gave a depth and thoughtfulness to the responses that we wouldn't have gotten with just individuals.

The users we talked to fell into two distinct categories with respect to their comfort levels with technology. The group session consisted of mostly Microsoft employees who are quite tech savvy and expect a lot from their technology solutions. The couples were not as tech savvy and so were less demanding of the app.

The key pieces of feedback received were the following:

- The app has to be easy enough to use to replace paper and pen for most people.
- Agreement on usefulness being mostly for families, especially the archetypal 'soccer mom'.
- Not convinced that it would be useful enough in a work environment given the lack of richness of metadata around tasks.
- Would be willing to pay a small amount for the app.
- Users are generally not aware of other apps that target group sharing of tasks.
- Multimedia additions to tasks don't seem to be an obvious benefit. Users could see how they might be useful but do not count them as essentials.
- Top features to add were:
    a. Due dates
    b. Sorting based on metadata (specifically due date)
    c. Ability to see all my tasks across groups
    d. Filtering for complete vs. active tasks
- Basic task view needs more detail to be useful (e.g. who it's assigned to).
- Group management is very similar to task management. Need more follow-up to see if this is a problem for users.
- WP7 platform and UX model is unfamiliar but most people weren't too scared by it.

We ended up putting more weight on the feedback received from the two couples rather than the group session. Given that our target audience is not a highly technical Microsoft employee, we didn't want to too heavily bias the app towards them.

Our modifications based on user feedback consisted of adding due dates to tasks. Because of the importance of this feature, we chose to implement it just before the project completed. Other items seemed to be much less critical.

PROJECT STATUS

SCHEDULE REVIEW

The following was our three week milestone schedule. We were successfully able to hit most of the items in each milestone. Comments are inline to address the overall completion of the milestone. Comments also contain information if there was something of note specific to an item within the milestone.

MILESTONE 1

Target Completion Date: 2/21
This milestone was completed successfully and on time. The items within this milestone helped define how we approached later milestone tasks. It also gave us a working design that we used when we did our usability study.

- Defining the user scenarios we hope to demonstrate and the workflow necessary for each distinct user scenario
- User interface screen review and lockdown
- Initial client-side flow design and implementation
- Mock up back-end using XML and persistent data on the phone
- Work on integration with Push Notification Service via self-host emulation

MILESTONE 2

Target Completion Date: 2/28
The main items within this milestone were completed successfully and on time. The only item that we were unable to achieve within this milestone was the relocation of the push notification implementation. This was not a high priority task for us because our focus was on the client-side rather than backend server side integration.

- Emulated Push Notification integration complete
- Refinements to client-side implementation necessitated by Push Notification integration and feedback on weekly assignments
- Relocation of Push Notification implementation from self-host emulation to live Windows Azure-based web service (time permitting)

MILESTONE 3

Target Completion Date: 3/7
This milestone was completed successfully and on time. We were able to take the feedback from the usability study and incorporate those into our demo during class. The application is in a more polished state and the workflow within the application makes sense and is very user friendly.

- Improvements of application flow to create improved usability
- Incorporation of feedback from user research where feasible
- Code clean-up, commenting, and visual polish

STATUS

CLIENT USER INTERFACE IMPLEMENTATION

The basic pages for login, viewing tasks and groups, location information, editing tasks, and settings are in place, as well as the application bar buttons and other links that tie them together. This has allowed us to get a good sense of the flow of the application as the user moves between views. This gave us a very good v1 user experience based on our architecture.

We have also completed the definition of a basic object representation for task information using XML, which is described in the Architecture Design section. This allowed us to flesh out our client-side spoof of a back-end, as well as define what data needs to be stored and passed back and forth between the application and the server. Since the main focus of our development was on the client-side user experience, the XML object representation was enough to allow us to fully expand on our v1 front-end design.

BACK-END IMPLEMENTATION

Since the primary focus of this class was to create an application on the phone, and not to develop a large scale back-end service to process data for the application and handle communication between devices, we came up with a compromise to implement a demonstrative “back-end” primarily through the phones and the emulation environment. This allowed us to fulfill the requirements of the project successfully, but also gave us the opportunity to demonstrate the collaborative nature of our application.

We were able to use the Microsoft Push Notification Service to supply change notifications and live tile updates to our clients via the emulator. It was suggested that we use SMS instead of push notifications; however, with the restrictions placed by Microsoft upon Windows Phone 7 API set, we determined that it would be more difficult and time-consuming to pursue this alternate approach. We completed a working prototype utilizing the service based on community-supplied training materials and then adapted it to the specific needs of our application. We successfully implemented the push notifications on the emulator, as demoed in our video, but, because of time constraints, were unable to move it to a live Windows Azure or IIS deployment. Had we been able to deploy the push notification service, we would have been able to demonstrate notifications sent to the app in a real-life phone environment.

In an effort to make our collaboration experience not restricted solely to the emulator, we intended to create a live version of the back-end component we implemented to generate push notifications. Windows Azure[12] was the likely candidate for this effort, as it provides a ready-made infrastructure. To this end, our group signed up for a program Microsoft offers full-time employees through the “MS Garage” initiative to promote Windows Azure; upon creation of our account, we were provisioned an Azure sandbox environment to learn and experiment in, as well as access to a large amount of internal documentation and help FAQs. Because Azure development was conditionally part of M2, we decided to re-focus our efforts on developing the v1 client-side user experience and were unable to get a back-end implementation working on Azure. Since our client-side user experience has been fleshed out, the Azure back-end is clearly part of the direction that would be taken for the development for the v2 experience.

LESSONS LEARNED

This was a great experience for the team to gain exposure to the Windows Phone 7 development environment. There are definitely some interesting challenges that we encountered because we were doing mobile development. One of the major ones was storing data and providing a service for communicating between devices and the data store. This part of the development for a mobile application is not the first part that one would think about, but is no less important than the actual client-side app.

We found that the development environment that Microsoft provides for Windows Phone 7 development was very easy to work with. There is plenty of documentation to help support learning the ins and outs of WP7. In short order, we were able to create a professional looking application that operated within the WP7 panorama layout design. This allowed us to concentrate on the larger problem of creating a base understanding for the design pattern of the application. Once we got a feel for the MVVM design pattern, the development went very quickly.

From the beginning there were a lot of unknowns for us. None of us had worked on WP7 or Azure development. Looking back, because of the time constraints of the class we may have originally over-estimated what we could get done. As we progressed and learned more about each environment, we were able to change the scope accordingly, staying more focused on the spirit of the project for the class – the client-side user experience.

On future projects, we know now to spend less time laying out initial application screens (with little code behind them) and focus more on understanding the data structures and design patterns for the entire application, as well as finding good templates and code samples to start with. The screen layouts were useful to have already pre-done when we finally did nail down the design patterns, but we budgeted and spent more time than ended up being necessary designing the screens in isolation from how the rest of the application would work. Once the design patterns were understood, the only blocker for completing the client implementation was available time remaining.

ACKNOWLEDGEMENTS

The Dialed In team would like to thank the following people for their participation in our usability study. They helped influence the final direction of the v1 client-side user experience. Many thanks for their time!

- Richard Cassan
- Greg Filpus
- Michael Friedman
- Bret Kiraly
- Adam Linkon
- Michael Loughry
- Ian Marshall

REFERENCES

1. Microsoft Corporation. (2011). *Track your to-do and task items*. Retrieved February 11, 2011, from Outlook 2007 Help and How-to: http://office.microsoft.com/en-us/outlook-help/track-your-to-do-and-task-items-HA001229302.aspx

2. Microsoft Corporation. (2011). *Share task folders with others*. Retrieved February 11, 2011, from Outlook 2007 Help and How-to: http://office.microsoft.com/en-us/outlook-help/share-task-folders-with-others-HA001229903.aspx

3. Microsoft Corporation. (2011). *Calendar: Keep track of your to-do list*. Retrieved February 11, 2011, from Explore Windows Live: http://explore.live.com/windows-live-calendar-to-do-list-using

4. Google. (2011). *Using Tasks*. Retrieved February 11, 2011, from Gmail Help: http://mail.google.com/support/bin/answer.py?hl=en&ctx=mail&answer=106237

5. Remember the Milk. (2011). *Learn More*. Retrieved February 11, 2011, from Remember the Milk: http://www.rememberthemilk.com/tour/

6. Microsoft Corporation. (2011, January 28). *Application Bar for Windows Phone*. Retrieved February 11, 2011, from MSDN Library: http://msdn.microsoft.com/en-us/library/ff431801%28v=VS.92%29.aspx

7. Microsoft Corporation. (2011, January 28). *Isolated Storage Overview for Windows Phone*. Retrieved February 11, 2011, from MSDN Library: http://msdn.microsoft.com/en-us/library/ff402541%28v=VS.92%29.aspx

8. Microsoft Corporation. (2011, January 28). *Location for Windows Phone*. Retrieved February 11, 2011, from MSDN Library: http://msdn.microsoft.com/en-us/library/ff431803%28v=VS.92%29.aspx

9. Microsoft Corporation. (2011, January 28). *Bing Maps Silverlight Control for Windows Phone*. Retrieved February 11, 2011, from MSDN Library: http://msdn.microsoft.com/en-us/library/ff941096%28v=VS.92%29.aspx

10. Microsoft Corporation. (2011, January 28). *Launchers and Choosers Overview for Windows Phone*. Retrieved February 11, 2011, from MSDN Library: http://msdn.microsoft.com/en-us/library/ff769542%28v=VS.92%29.aspx

11. Microsoft Corporation. (2011, January 28). *Push Notifications Overview for Windows Phone*. Retrieved February 11, 2011, from MSDN Library: http://msdn.microsoft.com/en-us/library/ff402558(VS.92).aspx

12. Microsoft Corporation. (2011). Retrieved February 11, 2011, from Windows Azure Platform: http://www.microsoft.com/windowsazure/